# Evolving Brain Structures for Robot Control *

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

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

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

**Abstract**

To study the relevance of recurrent neural network structures for the behavior of autonomous agents a series of experiments with miniature robots is performed. A special evolutionary algorithm is used to generate networks of different sizes and architectures. Solutions for obstacle avoidance and phototropic behavior are presented. Networks are evolved with the help of simulated robots, and the results are validated with the use of physical robots.

*appeared in: J. Mira, and A. Prieto (Eds.), "Bio-Inspired Applications of Connectionism" Proceedings IWANN'2001, Granada, Spain, June 13-15, 2001, Vol. II, *Lecture Notes in Computer Science*, **2085**, Springer-Verlag, 2001, pp. 410–417.

# 1 Introduction

Starting from the hypothesis that higher information processing or cognitive abilities of biological and artificial systems are founded on the complex dynamical properties of neural networks, an artificial life approach to evolutionary robotics [4] is investigated. Complex dynamical properties of small neural networks, called neuromodules for obvious reasons, are in general caused by a recurrent coupling structure involving excitation and inhibition. Thinking about these neuromodules as functionally distinguished building blocks for larger brain-like systems, the (recurrent) coupling of these non-linear subsystems may lead to an emergent behavior of the composed system.

Because the structure of neuromodules, with respect to adaptive behavior of autonomous agents, is hard to construct [1], an evolutionary algorithm to develop neuromodules as well as the couplings between these subsystems is used. This algorithm is called $ENS^3$-algorithm for *evolution of neural systems by stochastic synthesis*. The $ENS^3$ has already been tested successfully on nonlinear control problems [6]. Furthermore, autonomous systems acting in a sensori-motor loop, like simulated or real robots, are an appropriate tool for studying the development of embodied cognition [7].

For the experiments reported in this paper the miniature Khepera robots [3] as well as the Khepera simulator [2] are used. With respect to a desired robot behavior, the neural networks are evolved with the help of the Khepera simulator. Those generating an excellent performance in the simulator are then implemented in the physical robot and tested in various physical environments. Finally, the structure of successful networks can be analysed. It turned out, that evolved networks with an outstanding performance can be comparatively small in size, making use of recurrent connections.

The $ENS^3$-algorithm is applied to networks of standard additive neurons with sigmoidal transfer functions. The algorithm sets no constraints, neither on the number of neurons nor on the connectivity structure of a network, developing network size, architecture, and parameters like weights and bias terms simultaneously. Thus, in contrast to genetic algorithms, it does not quantize network parameters and it is not primarily used for optimizing a given architecture. The algorithm will be outlined in section 2. For the solution of extended problems (more complex environments, more complex sensori-motor systems, more complex survival conditions, etc.) the synthesis of evolved neuromodules forming larger neural systems can be achieved by evolving the coupling structure between functionally distinguished modules. This may be done in the spirit of co-evolution of interacting species. We suggest that this kind of evolutionary computation is well suited for evolving neural networks, especially those with recurrent connectivity and dynamical

features.

## 2 The $ENS^3$ evolutionary algorithm

For the following experiments we use networks with standard additive neurons with sigmoidal transfer functions for output and internal units, and simply buffers as input units. The number of inputs and outputs is chosen according to the problem; that is, it depends on the number of involved sensors and the necessary motor signals. Here networks will have to drive only the two motors of the Khepera. Thus, we use *tanh* as neural transfer function, setting bias terms to zero, to provide positive and negative signals for forward/backward operations of the motors. 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 units are allowed.

To evolve appropriate networks we consider a population $p(t)$ of $n(t)$ neuromodules undergoing a variation-evaluation-selection loop, i.e.

$$p(t+1) = S\,E\,V\;p(t)\;.$$

The *variation operator* $V$ is realized 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 given in terms of a fitness function. This function usually has two types of terms: those defining the performance with respect to a given behavior task (system performance), and terms related to internal network properties like network size and the number of connections. After evaluating the fitness 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* $S$. It performs the differential survival of the varied members of the population according to evaluation results. During repeated passes through the variation-evaluation-selection loop the average performance of the populations will increase. If a satisfactory system performance is achieved, the evolution process will generate smaller networks of equal system performance, if terms in the fitness function are set appropriately.

# 3 Evolved networks for Khepera control

The goal of the following experiments is to generate a specific robot behavior like obstacle avoidance or phototropism. We use an average population size of 30 individuals. The time interval for evaluating these individuals was set to 2000 simulator time steps. There was also a stopping criterion for the evaluation of an individual: It was terminated when bumping into an obstacle. Furthermore, we influence the size of resulting networks by adding cost terms for neurons and connections to the given fitness functions.

## 3.1 The first experiment: Obstacle avoidance

In a first experiment a network which allows the simulated Khepera robot to move in a given environment without hitting any obstacle present in this environment is evolved. The final goal is to obtain networks, evolved in a simulated world, which produce a comparably successful behavior, when controlling the physical robot in its very different environment.

For solving this task, the six infra-red proximity sensors in the front, and the two in the rear of the robot provide the input to the network. To control the two motors of the Khepera, the signals of the two output neurons are used. Thus, initially the individual neuromodules have only eight linear input neurons as buffers and two nonlinear output neurons.

To reach the goal fitness function $F$ is introduced which simply states: For a given time T go straight ahead as long and as fast as possible. This is coded in terms of the network output signals $out_1$ and $out_2$ as follows. First the quantities $m_1$ and $m_2$ are defined as follows:

$$\text{if} \quad out_i \leq 0 \quad \text{then} \quad m_i = 0, \quad \text{else} \quad m_i = out_i, \quad i = 1, 2 \; .$$

Then, the fitness function is given by

$$F := \sum_{t=1}^{2000} \alpha \cdot (m_1(t) + m_2(t)) - \beta \cdot | \, m_1(t) - m_2(t) \, | \; , \tag{1}$$

with appropriate parameters $\alpha$ and $\beta$.

Starting with a simple environment (with less walls and obstacles than the environment shown in Fig. 2) it takes around 100 generations (depending on the parameter settings of the evolutionary program) to get individuals having satisfactory fitness values. Although generating a comparably good robot behavior, the corresponding neural networks can differ in size as well as in their connectivity structures.

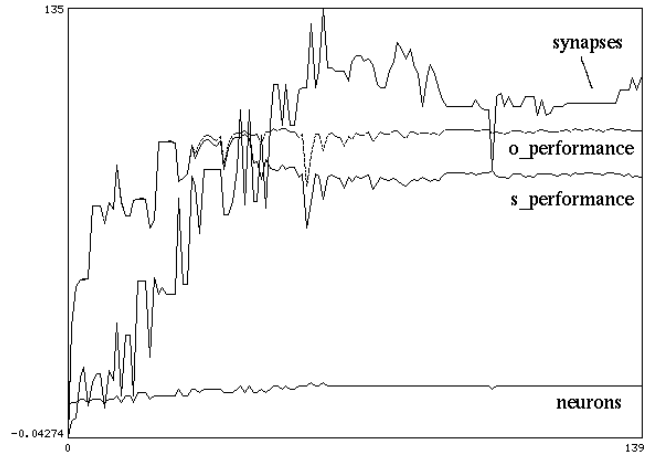


Figure 1: Performance of simulated robots and the development of corresponding network size during the first 139 generations of an evolutionary process.

The development of brain size (number of neurons and number of connections) of a typical evolutionary process can be followed in Fig. 1. At the beginning robots do not move or are spinning around having only one wheel active, but already after ten generations the first robot is moving slowly on a straight line. After thirty generations the fittest robots are exploring the accessible space, moving faster on straight lines and turning when near a wall. During this phase networks are still growing in general. The irregular development of network size corresponds to the different initial conditions from which robots in every generation have to start; they are more or less difficult to cope with. Then, around generation 45 the costs for neurons and synapses are increased to keep the size of the networks small. Therefore the curves for the output performance and the system performance are now splitting. At around generation 60 the number of hidden neurons of best networks stabilizes around 6, with networks having around 110 synapses. A further increase in costs for the network size is able to reduce the number of neurons and synapses without decrease in the performance of the controllers.

To make the simulator solutions more robust, in the sense that they can control the real robot in its very different physical environment equally efficient, the simulator environments gradually changed having more obstacles or walls, including for instance walls at 45 angles. An example environment is shown in Fig. 2a, together with a typical path of a simulated robot in this environment. The paths for the left and the right wheel start at the

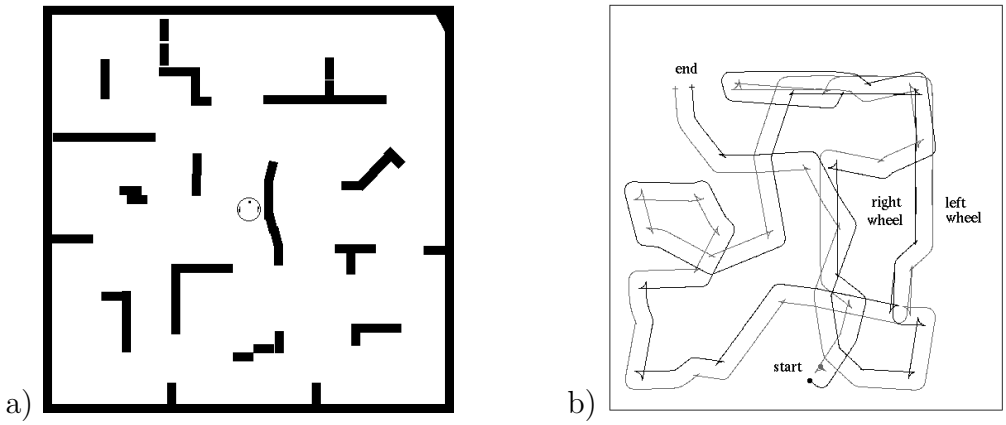


Figure 2: a.) A simulator environment and b.) a robot path in this environment for 5000 time steps.

filled circles and end at the cross marks. It can be seen, that the robot turns left as well as right in different situations, achieving this by turning the corresponding wheel backwards for a short while.

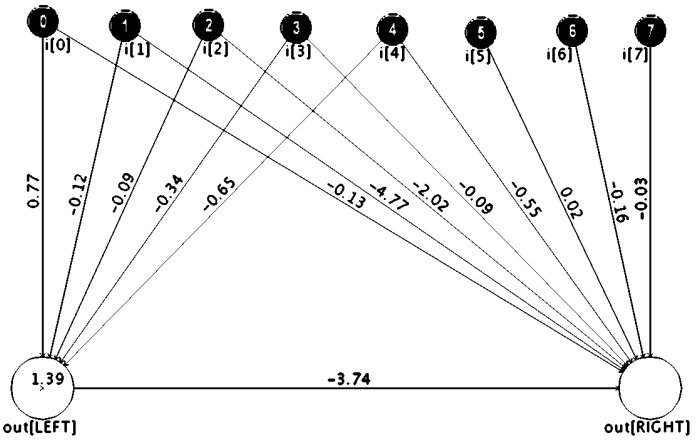


Figure 3: The simplest evolved network which was able to generate an effective obstacle avoidance behavior for the real robot.

The most simple solution, which also generates a very good behavior of the real robot, is shown in Fig. 3. It uses no internal neurons, but only direct connections from inputs to the output neurons. What is remarkable about this solution, is its output neuron configuration. The neuron driving the left motor has a self-inhibitory neuron which also inhibits the second output neuron. Furthermore, it should be noted, that not all sensors are connected to the left or to the right output neuron, that the rear proximity sensors are

connected only to right output neuron $out[0]$, and that the whole connectivity structure is highly asymmetric with respect to the left/right symmetry of the sensor and motor configuration. Remarkably, most of the connections are inhibitory. It is well known, that for a single neuron with positive self-connection larger than 1 there is an input interval over which a hysteresis phenomena [5] (flipping between two stable states) can be observed. It seems that in the shown network this effect is effectively used for instance to get out of situations like 45 angles between walls.

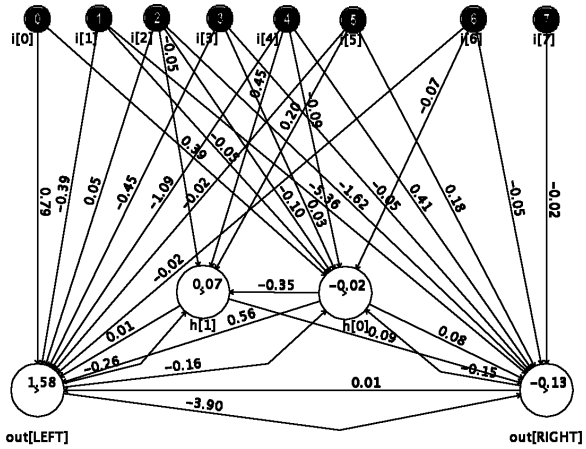


Figure 4: A more complex network generating effective obstacle avoidance for the real robot.

A second solution shown in Fig. 4 uses two hidden neurons and several closed signal loops. All neurons (hidden and output) use self-connections, and loops are of different length - like the 2-loop between the output neurons, the 3-loop involving neurons $out[0]$, $h[0]$ and $h[1]$, and the 4-loop involving $out[0]$, $h[0]$, $h[1]$ and $out[1]$. Furthermore, loops are even or odd, that is, the number of inhibitory connections in a loop is even or odd. The generated behavior in the real robot is slightly smoother than the one generated by the first network. The robot not only finds its way out of 45 angle walls but also maneuvers out of a narrow blind alley.

The general impression of the robots behavior is of course not only obstacle avoidance but that of an exploratory behavior: letting the robot move in its physical environment after a while it will visit almost all areas within reach, moving through small openings in the walls, wandering through narrow corridors and even coming out of dead ends. This of course cannot be achieved by pure wall following behaviors, as it is usually learned by robots.

## 3.2 The second experiment: Light seeking

For the second experiment in addition to the 8 proximity sensors, the 8 ambient light sensors of the Khepera are used; i.e., we now have 16 sensor inputs. The goal of this experiment is to find a light source as fast as possible and to stay there (eating). Because the first experiment already provided networks generating an exploratory behavior, for the initial population it is reasonable to use one of these network solutions. The additional light sensors are of course not yet connected. During the evolution process additional neurons will connect to these new inputs while the initial network configuration may change.

To find an appropriate solution the following fitness function is applied stating: For a given time and a given environment "eat" as much light as possible. This is coded as follows

$$F := \sum_{t=3D1}^{2000} \alpha \cdot in_f(t) + \beta \cdot | \, in_f(t) - in_r(t) \, | \, , \tag{2}$$

where $\alpha$ and $\beta$ are appropriate parameters, and $in_f$ and $in_r$ are given in terms of the inputs $i[0]$, ..., $i[7]$ to the additional light sensors by

$$in_f := i[0] + i[2] + i[3] + i[5], \quad in_r := i[6] + i[7] \, .$$

Thus, for determining the fitness only four of the six front light sensors are used, and the proximity sensors are not evaluated at all.

Again, the $ENS^3$-algorithm is applied to the simulated robots with a few light sources now spread over the environments used also in the first experiment. As before, we gradually make the boundary conditions more complex; i.e., more walls and obstacles are introduced while reducing the number of light sources.

After having evolved networks for the simulated robots which seemed to generate the desired behavior in several different environments these networks are tested for the real robot. In fact, also the physical robot "looked" for a light source and got there straight when one was found. It even followed the light source, a small lamp, when this was moved by hand. The generated behavior seemed to be robust also in the sense that it was not much influenced by changing light conditions. Again there were reasonable solutions of different network size and structure. One interesting solution is shown in Fig. 5, where, for greater clarity, Fig. 5a shows the connections coming from proximity sensors and Fig. 5b those coming from the light sensors. It can be seen, that both types of input signals address all hidden and output neurons; i.e., both types of information are processed in one and

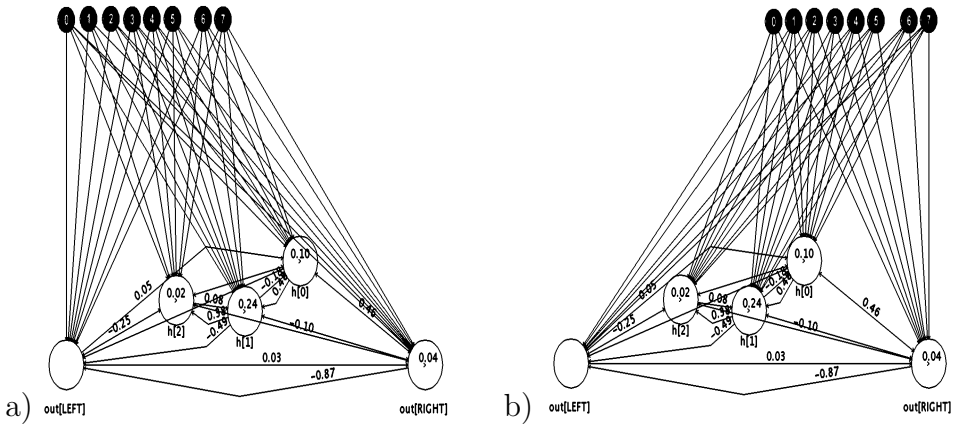


Figure 5: a.) Connections of the proximity sensors and b.) of the ambient light sensors to the hidden and output neurons of an evolved network for phototropic behavior.

the same network structure simultaneously. Furthermore the massively recurrent structure of this network is obvious, including also self-connections. Especially, the output neurons are again recurrently connected.

# 4 Final Remarks

The presented method to evolve neural networks for robot control turned out to be effective in two different directions. On one hand, the developed networks produced a robust robot behavior in the sense that one and the same network was not only able to cope equally good with different environments in the simulator, but turned out to have an excellent performance also when implemented in the physical Khepera and tested under different environmental boundary conditions. On the other hand, the $ENS^3$-algorithm produced a variety of recurrent neural networks, which can be analyzed with respect to their internal dynamical properties. This can reveal relations between these properties and the generated robot behavior. The mentioned hysteresis effect, which seemed to be responsible for effective turning in dead ends, may serve as a first example.

Our strategy is to progress from simple to more complex robot behaviors by evolving larger networks starting from initial populations of smaller functionally distinguished neuromodules. In principle there are two different techniques to achieve this. One method, called *module expansion*, starts from already evolved networks which can solve a subtask. Evolution will then add new neurons and connections to these, probably fixed, structures. It was

applied in the experiment described above, where phototropic behavior was created from an existing exploration ability.

The second method may be called *module fusion*: two evolved functionally distinguished modules, with fixed architectures, are chosen for the initial population and the evolution process generates only an appropriate coupling structure to accomplish a more extensive behavior task. Here, of course, emergent properties have to be expected and are desired: The resultant behavior may not be due just to the "sum" of the basic abilities, but may be of a very different quality.

# References


[1] Husbands, P., and Harwey, I. (1992) Evolution versus design: Controlling autonomous robots, in: *Integrating perception, planning and action: Proceedings of the Third Annual Conferences on Artificial Intelligence*, IEEE Press, Los Alamitos.

[2] 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`

[3] Mondala, F., Franzi, E., and Ienne, P. (1993), Mobile robots miniturization: a tool for investigation in control algorithms, in: *Proceedings of ISER' 93*, Kyoto, October 1993.

[4] Nolfi, S., and Floreano, D. (2000) *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines* 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. (1998), Evolving neurocontrollers for balancing an inverted pendulum, *Network: Computation in Neural Systems*, **9**, 495–511.

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