Max-Planck-Institut
**für Mathematik
in den Naturwissenschaften
Leipzig**

Evolving structure and function of neurocontrollers

by

Frank Pasemann

Preprint-Nr.: 27 1999

Evolving Structure and Function of Neurocontrollers*

F. Pasemann[‡], U. Dieckmann[†], and U. Steinmetz[‡]

[‡]Max-Planck-Institute for Mathematics in the Sciences
Inselstr. 22-26, D-04103 Leipzig, Germany

[†]International Institute for Applied Systems Analysis
ADN, A-2361 Laxenburg, Austria

Abstract

The presented evolutionary algorithm is especially designed to generate recurrent neural networks with non-trivial internal dynamics. It is not based on genetic algorithms, and sets no constraints on the number of neurons and the architecture of a network. Network topology and parameters like synaptic weights and bias terms are developed simultaneously. It is well suited for generating neuromodules acting in sensorimotor loops, and therefore it can be used for evolution of neurocontrollers solving also nonlinear control problems. We demonstrate this capability by applying the algorithm successfully to the following task: A rotating pendulum is mounted on a cart; stabilize the rotator in an upright position, and center the cart in a given finite interval.

*in: Proceedings, Concress on Evolutionary Computation (CEC'99), Washington, July 6-9, 1999, IEEE Press.

1 Introduction

The combined application of neural network techniques and evolutionary algorithms turned out to be a very effective tool for solving an interesting class of problems (for a review see e.g. [1], [8], [11]). Especially, in situations where a task involves dynamical features like generation of temporal sequences, recognition, storage and reproduction of temporal patterns, or for control problems requiring memory to compute derivatives or integrals, other learning strategies are in general not available.

The *ENS*^{3}-algorithm (*evolution of neural systems by stochastic synthesis*) outlined in section 2 is inspired by a biological theory of coevolution. Based on a behavior-oriented approach to neural systems, the algorithm originally was designed to study the appearance of complex dynamics and the corresponding structure-function relationship in artificial sensorimotor systems; i.e. the systems to evolve are thought of as "brains" for autonomous robots or software agents. The basic assumption here is that in "intelligent" systems the essential, behavior relevant features of neural subsystems, called neuromodules, are due to their internal dynamical properties which are provided by their *recurrent* connectivity structure [6]. But the structure of such nonlinear systems with recurrences can in general not be designed. Thus, the main objective of the *ENS*^{3}-algorithm is to evolve an appropriate structure of (recurrent) networks and not just to optimize a given (feedforward) network structure. It is applied to networks of standard additive neurons with sigmoidal transfer functions and sets no constraints on the number of neurons and the architecture of a network. In fact, it develops network topology and parameters like weights and bias terms simultaneously on the basis of a stochastic process.

In contrast to genetic algorithms, which are often only used for optimizing a specific feedforward architecture [8], [11], it does not quantize the network parameters like weights and bias terms. With respect to algorithms like, for instance, EPNet [12], it does not include an individual "learning" procedure, which exists naturally only for feedforward networks and problems where an error function or reinforcement signals are available.

For the solution of extended problems (more complex environments or sensorimotor systems) the synthesis of evolved neuromodules forming larger neural systems can be achieved by evolving the coupling structure between modules. This is done in the spirit of coevolution of interacting species. We suggest that this kind of evolutionary computation is better suited for evolving neural networks than genetic algorithms.

In [7] we reported on tests of the algorithm, applying it to the pole-balancing problem that usually serves as a benchmark problem for trainable controllers [5]. Of course, the inverted pendulum is one of the simplest inherently unstable systems, and balancing it under benchmark conditions is mainly in the domain of linear control theory. Stabilizing a pendulum which is free to rotate, and initially may be pointing downward, is therefore a more challenging nonlinear control problem [2]. Here, stabilization of an unstable stationary state, and destabilization of a stable stationary state have to be realized by one controller [9], [10]. In section 3 we will show that this problem is easily solved by evolved neural network solutions if the controller has access to the full phase space information. Two "minimal" solutions of the feedforward type are presented, although also recurrent networks were generated by the algorithm. If input signals are reduced to only the cart position and the pole angle the problem can not be solved by a pure feedforward structure because differentiation has to be involved in the processing. We present a parsimonious 2-input solution to the swinging-up problem in section 3.3, which utilizes a recurrent network structure.

Using continuous neurons for the controllers, different from many other applications, our approach does not make use of quantization, neither of the physical phase space variables nor of internal network parameters, like synaptic weights and bias terms, or output values of the neurons. Section 4 gives a discussion of the results.

2 The ENS^3-Algorithm

To start the algorithm one first has to decide which type of neurons to use for the network. We prefer to have 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. $p(t+1) = S\,E\,V\;p(t)$. 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 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. It realizes 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.

To demonstrate the basic functioning of the ENS^3-algorithm we will discuss its application to a toy control problem (balancing a rotator on a cart), which involves the handling of conflicting properties: stabilizing an unstable stationary state (balancing) and de-stabilizing a stable one (swinging-up).

2.1 Setting the Problem

The problem to solve is given here as follows: A rotating pendulum is mounted on a cart that can move on a 1-dimensional interval. The controller has to bring the pendulum into the upright position and then has to balance it as long as possible. At the same time, interval boundaries have to be avoided, and the cart has to be centered. The control signal is given by the force on the cart. Because we use neurons with sigmoidal transfer functions the force applied to the cart varies continuously between $-10 < F < 10$ $[N]$. The cart is bound to move in the interval $-2.4 < x < 2.4$ $[m]$. The initial position θ_0 of the pendulum can be anywhere on the circle with initial velocity $\dot{\theta}_0 = 0$. The cart starts from positions $-1.0 < x_0 < 1.0$ with zero velocity $\dot{x}_0 = 0$.

The equations for the physical system under control are given by

$$\ddot{\theta} = \frac{g\,sin\,\theta - \frac{cos\,\theta}{m_c+m}(\,F + m\,l\,\dot{\theta}^2\,sin\,\theta\,)}{l\,[\frac{4}{3} - \frac{m\,cos^2\,\theta}{m_c+m}]}\ ,$$

$$\ddot{x} = \frac{F + m\,l\,(\,\dot{\theta}^2\,sin\,\theta - \ddot{\theta}\,cos\,\theta\,)}{m_c + m}\ ,$$

where $g = 9.81\,\mathrm{ms}^{-2}$ denotes gravitational acceleration, $m_c = 1.0\,\mathrm{kg}$ and $m = 0.1\mathrm{kg}$ mass of cart and pendulum, respectively, $l = 0.5\,\mathrm{m}$ half length of pendulum, and F denotes the force applied to the cart. We use no friction terms because we found these have no interesting effect on the evolution process or network capabilities. The dynamics of cart and pendulum are computed by using Euler discretization of these equations with time step $\Delta = 0.01\,\mathrm{s}$.

For the neurocontroller we use the standard additive neuron model with sigmoidal transfer function σ. A termination signal is given after a time $t > t_{max}$. The highest fitness of an individual network corresponds to the minimum of the corresponding cost function C; it has the general form

$$C = c_1 \cdot P_\theta + c_2 \cdot P_x + c_3 \cdot N_n + c_4 \cdot N_s + c_5 \cdot I_F\ ,$$

where the constants $c_1, \ldots, c_5$ are all positive. It takes into account the pendulum's integrated deviation from the upright position P_θ, and the cart's integrated deviation from the zero position P_x (they define the problem), costs $c_3 \cdot N_n$ for each neuron and for each connection $c_4 \cdot N_s$ (to select for parsimonious network architectures); N_n and N_s denote the number of neurons and synapses, respectively. Furthermore, the applied force I_F, integrated over the last 20 seconds of each trial, can also be added. This will optimize the applied force to balance the pendulum by minimizing oscillations of the cart. The cost factors $c_1, \dots, c_5$ have to be appropriately chosen, and their choice will influence the properties of the evolutionary process considerably. During a run of the *ENS*^{3}-algorithm these cost factors can be varied on-line.

We distinguish between two classes of controllers. One, called *t-class*, uses additive units with anti-symmetric transfer function $\sigma(x) = \tanh(x)$, the other one, the *s-class*, uses the strictly positive transfer function $\sigma(x) = (1 + e^{-x})^{-1}$. The first class of controllers needs only one output neuron providing a force

$$F = 10 \cdot \tanh(a_i)[\mathrm{N}] \, , \tag{1}$$

where a_i denotes the activity of the output unit i. The s-class needs two output units, i and $i+1$, giving a force

$$F = 10 \cdot (\sigma(a_i) - \sigma(a_{i+1}))[\mathrm{N}] \, . \tag{2}$$

3 Evolved neurocontrollers

For the following evolved solutions we used an average number of 50 individuals in each population. A successful run had about 10000 generations. Because the *ENS*^{3}-algorithm is primarily designed to study theoretical aspects of modularized recurrent networks, we were not concerned about statistics or computation time. The required computing time depends strongly on the parameter settings - their optimal values in the context of a given problem are not known from the beginning - and on the design of an appropriate fitness function. Parameters of the algorithm are, for instance, the probabilities for insertion and deletion of neurons and connections, and for alteration of bias and weight terms; furthermore, the costs for neurons, for connections, and for the applied force, the steepness of the selection function, the average population size, the maximal time to solve the problem (stop criterion), and the like.

It should be mentioned that all the probabilities (for insertion/deletion of neurons, insertion/deletion of connections, variations of weights and bias terms, etc.) all can be set separately. For the evolution of the networks described below the following typical parameter values of the algorithm had been used: The probabilities for inserting and deleting a neuron where both set to 0.05, the probability to set a connection from an inserted neuron to existing neurons where set to 0.5. Probabilities for inserting and deleting synapses where both set to 0.1, and the probabilities for varying connection strengths and bias terms where set to 0.3. But often the possibility to vary these parameters on-line during the evolutionary process has been used. To obtain parsimonious network structures one has to “balance” the probabilities for inserting neurons and for setting its connections.

During intermediate states of the evolutionary process the fittest modules may become quite large - more than 20 neurons and 60 synapses - and network size and architecture are often varied. Finally there appear smaller modules with equally good or better performance. Some of the "minimal" solutions found by the ENS^3-algorithm are described in the following.

3.1 A t-class Controller with Four Inputs

As sensor signals we choose the full set of state variables x, θ, $\dot{x}$, $\dot{\theta}$ of the physical system. The corresponding four input units then receive the signals:

$$in_1 := x/2.4\ ,\ in_2 := \theta/\pi\ ,\ in_3 := \dot{x}/2.4\ ,\ in_4 := \dot{\theta}/\pi\ . \tag{3}$$

The output unit 5 of a t-class controller provides the force F applied to the cart according to equation (1).

Among a family of larger modules, the ENS^3-algorithm came up with the following minimal solution: The architecture of this controller w^1 is shown in figure 1 and its weights are given as follows:

$$w_5^1 = (0,\ -7.56,\ -20.0,\ -11.88,\ 0,\ 0,\ -9.66)$$
$$w_6^1 = (0,\ -1.65,\ -17.83,\ -3.19,\ -4.88,\ 0,\ 0)$$

where $w_i = (w_{i0}, w_{i1}, \ldots, w_{in})$ denotes the weight vector of its neuron i, with w_{i0} denoting the bias term of unit i.

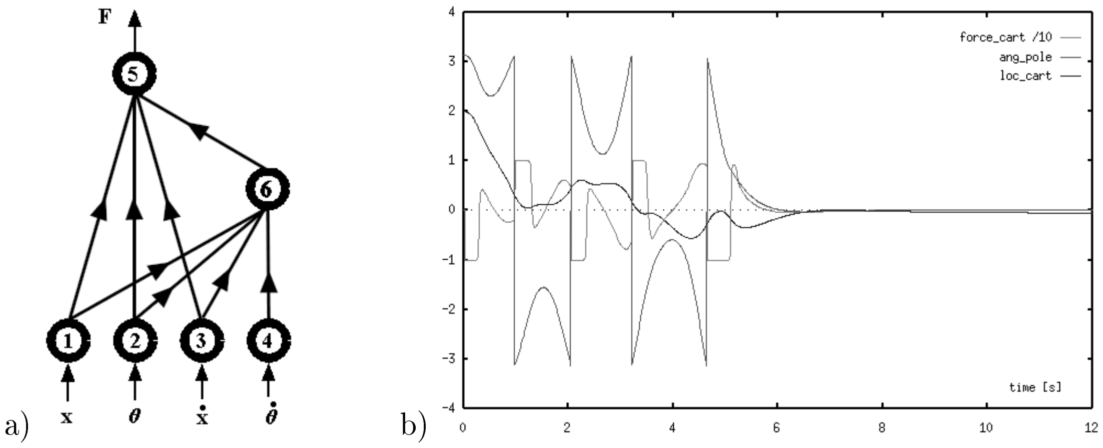


Figure 1: a.) A minimal 4t-class solution w^1 and b.) its effective control: $x(t)$, $\theta(t)$, and $F(t)$ starting from $x_0 = 2.0$ and $\theta_0 = \pi$.

Although this module has a very simple feedforward structure, tests revealed that it solves the problem for all initial conditions $-2.0 < x_0 < 2.0$ and $-\pi < \theta_0 < \pi$ in less than 10 seconds. This is demonstrated for instance in figure 1b where cart position x, angel θ and the applied force F are given as functions of time

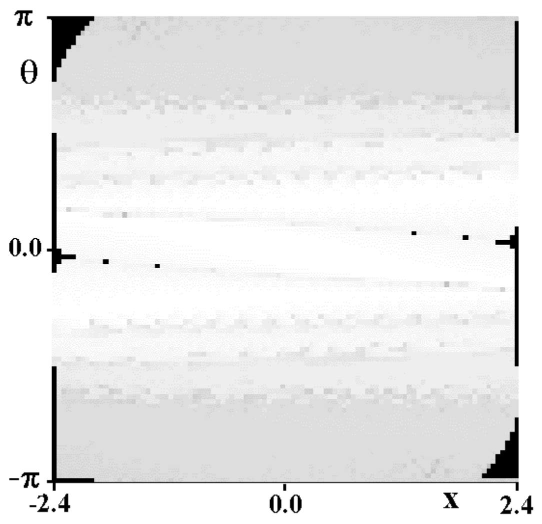


Figure 2: Performance of controller w^1 on (x, θ)-initial conditions: White=100%, black=0%.

t. Starting with initial conditions $x_0 = 2.0$ (cart close to boundary at $x = 2.4$), $\theta_0 = \pi$ (pendulum pointing down), and $\dot{x}_0 = \dot{\theta}_0 = 0$ we observe that the controller needs only three swings to get the pendulum into the upright position, and then it balances the pendulum by centering the cart at the same time. That the controller w^1 has a comparable good performance for almost all initial conditions is demonstrated in figure 2 where the (x, θ)-space is divided into 100x100 squares: The grey scale represents the output performance of the controller during the first 12 seconds; with black representing 0% and white= 100%; start velocities are $\dot{x} = \dot{\theta} = 0$. The rotator is balanced for almost all (x, θ)-positions of the cart-rotator system (not black). Grey shades correspond mainly to small cart oscillations which still occur after some seconds if the rotator had to be swung up from an almost downward position. Black squares indicate failure before the end of the maximal evaluation time $t_{max} = 12$ seconds.

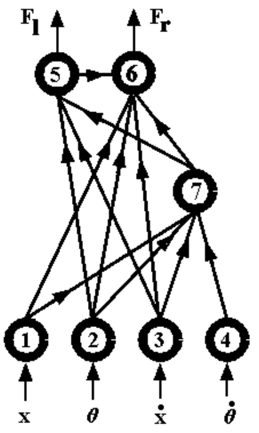
The module in figure 1 displays already an interesting feature: it can be understood as composed of two submodules. The structure of the one given by neuron 6 with its four inputs is known as that of a pole balancing module [7]. The module given by neuron 5 with its three inputs swings up the pole from downward positions when isolated. They are coupled through the connection w_{56}.

3.2 An s-class Controller with Four Inputs

Sensor signals are again given by equation (3). The force on the cart is applied according to equation (2) for output units 5 and 6. Again several neurocontrollers emerged during the evolution process and one of the minimal examples, the controller w^2, is shown in figure 3 with weights given by

$$\begin{aligned} w_5^2 &= (8.2,\ 0,\ -30.46,\ -8.38,\ 0,\ 0,\ 0,\ -13.34) \\ w_6^2 &= (-6.88,\ 21.47,\ 12.09,\ 12.45,\ 0,\ -12.2,\ 0,\ 28.43) \\ w_7^2 &= (0.24,\ -2.15,\ -40.0,\ -5.23,\ -7.46,\ 0,\ 0,\ 0) \end{aligned}$$

where again $w_i = (w_{i0}, w_{i1}, \ldots, w_{in})$ denotes the weight vector neuron i, with w_{i0} the bias term.

a) 

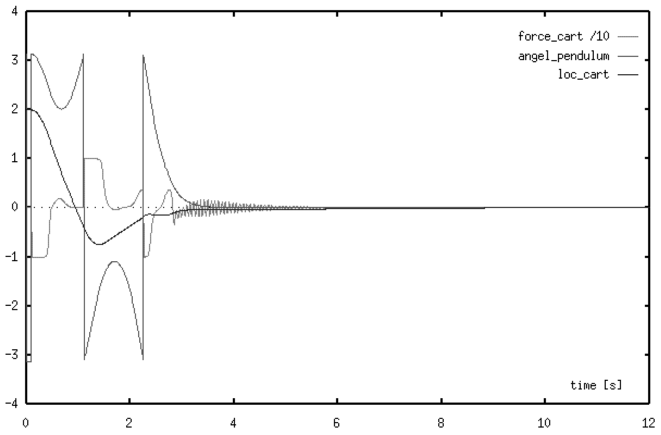
b) 

Figure 3: a.) A 4s-class solution w^2 and b.) its effective control: $x(t)$, $\theta(t)$, and $F(t)$ starting from $x_0 = 2.0$ and $\theta_0 = \pi$.

Also this neurocontroller uses only one internal neuron and a no recurrent connections to solve the problem. The output neuron 6 gets a "lateral" connection from output neuron 5. Figure 3b reveals that it is even faster than the t-class controller. It needs only two swings to get the pendulum into the upright position, starting from $x_0 = 2.0$ and $\theta_0 = \pi$. Stabilizing the pendulum and centering the cart is done with a small oscillating force signal. The origin of these oscillation is not given by an internal oscillator of the neural structure, but it results from the feedback loop via the environment. The performance of controller w^2 on other initial conditions is roughly comparable to that of w^1, as is displayed in figure 4. The main difference is that the performance of w^2 is not symmetric on (x, θ)-space because the bias terms of its output units are not fine tuned, to give zero force for zero inputs.

Again, this system can be viewed as composed of two submodules: The module given by neuron 7 with its two inputs acts on the module given by the connections from the inputs to the output neurons plus the connection w_{65}.

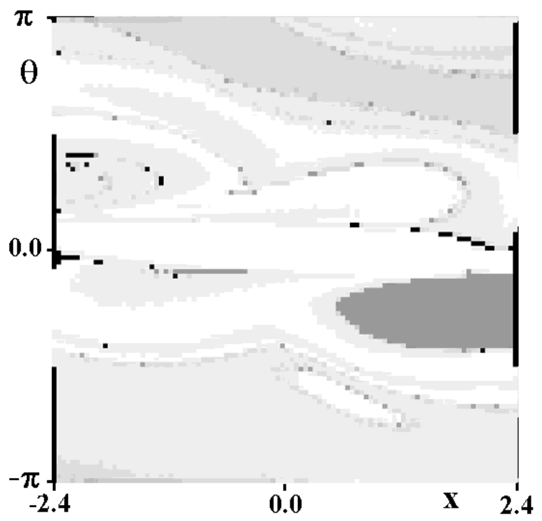


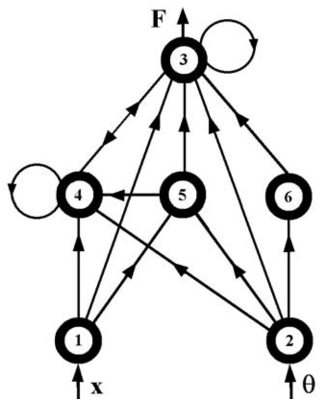
Figure 4: Performance of controller w^2 on (x, θ)-initial conditions: White=100%, black=0%.

4 A t-class Controller with Two Inputs

Reducing the input signals to only cart-position x and angle θ makes the the problem more sophisticated. The control module now has to compute derivatives, and therefore evolving recurrent connections have to be expected. The claim is, that there exists no feedforward network which is able to solve the task (and, to our knowledge, there is no solution of this kind described in the literature). Solving the problem under these restrictive input conditions seems to be a much harder problem. In fact, we can not present an evolved neurocontroller, which acts as successfully for all initial conditions of the physical system as, for instance, controller w^1. But it was quite easy to evolve a controller which is able to solve the *swinging-up problem* [10], i.e. starting the cart-rotator system from initial conditions $x_0 = 0$, $\theta_0 = \pi$. The architecture of this controller w^3 is shown in figure 5, and its weights are given as follows:

$$\begin{aligned}
w_3^3 &= (-0.22,\ -9.86,\ 0.36,\ -0.79,\ 1.42,\ 4.59,\ -4.49) \\
w_4^3 &= (-0.45,\ -1.31,\ -6.49,\ -7.48,\ -0.36,\ -1.2,\ 0) \\
w_5^3 &= (-0.01,\ 3.44,\ 26.7,\ 0,\ 0,\ 0,\ 0) \\
w_6^3 &= (-0.05,\ 0,\ 10.41,\ 0,\ 0,\ 0,\ 0)
\end{aligned}$$

Figure 5b shows that controller w^3, although having only two inputs, is able to swing up the rotator from $x_0 = 0$, $\theta_0 = \pi$, $\dot{x}_0 = \dot{\theta}_0 = 0$ and to balance it in less than five seconds. But then balancing is achieved by a strong periodic force signal to the cart. This spoils of course the overall performance of the controller.

a) 

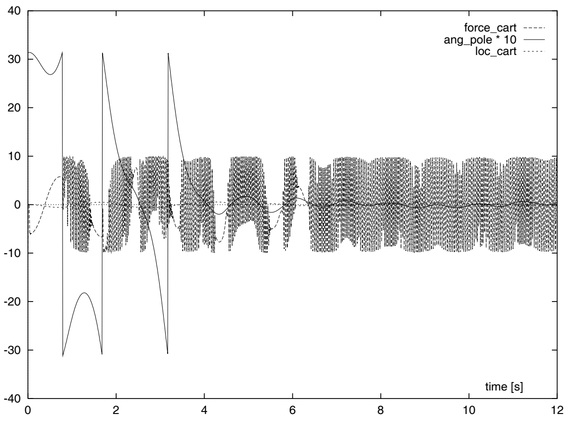
b) 

Figure 5: a.) The 2t-controller w^3 solving the swinging-up problem, and b.) cart position and pole angle under its action, starting from $x_0 = 0$, $\theta_0 = \pi$.

The performance of w^3 during the last 10 seconds of the maximal evaluation time $t_{max} = 20$ [s] on all of the (x, θ)-initial conditions is displayed in figure 6. It shows that there are some isolated start positions for which the controller is able to solve the task (white squares). Especially the central light region indicates that the controller acts there as a pole balancer. Black dots again represent positions where the controller fails before the maximal evaluation time t_{max} is reached. For grey shaded initial conditions the controller keeps the pendulum swinging or rotating. Black dots correspond to situations where the cart hits the interval boundary before the end of the maximal evaluation time t_{max}.

As expected, the controller w^3 uses recurrent connections to solve the problem: Two self-connections, w_{33} and w_{44}, and the loop w_{34}, w_{43}. In fact, the module composed of neurons 3 and 4 has complex dynamical properties; for accessible (stationary) inputs it has quasiperiodic attractors as well as attractors of high periodicity. Several delay lines for input signals along neurons 5 and 6 probably do also contribute to the successful operation of the controller.

5 Conclusions

We have demonstrated that the ENS^3-algorithm can be applied successfully to a challenging nonlinear control problem like balancing a rotating pendulum. The evolved network solutions for the four input problem are remarkably small in size (compare e.g. with network solutions derived in [10]). They solve the problem very effectively by getting the pendulum in upright position, stabilizing it and centering the cart in less then 10 seconds, without hitting interval boundaries, and they do that for almost all initial conditions from intervals $-\pi < \theta_0 < \pi$ and $-2.0 < x < 2.0$. Because they have full access to physical phase space variables, they do not need recurrences to compute derivatives. Remarkable is that both

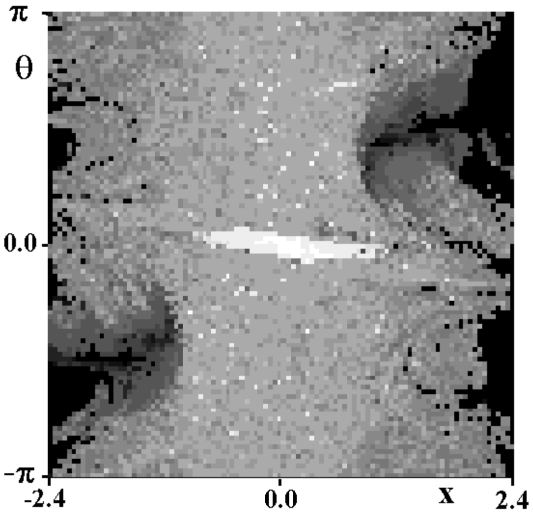


Figure 6: Performance of controller w^3 on (x, θ)-initial conditions: White=100%, black=0%.

four input solutions w^1 and w^2 can be described as a composition of interacting subsystems - one is more or less responsible for the destabilization of the stable stationary state, the other for the balancing, i.e. the stabilization of the unstable stationary state.

As for the pole-balancing problem discussed in [7], the problem becomes much harder if controllers get only information about cart position and angle of the pendulum. Then the problem is no longer solvable with a pure feedforward network, because differentiation now has to be done by the controller itself. The evolved controller w^3 presented in section 3.3 swings-up the pendulum from the downward position, i.e. $x_0 = 0$, $\theta_0 = \pi$, but it does not operate with comparable success on all other initial conditions. We suggest, that the type of fitness function used for this problem or the chosen combination of cost factors is not yet appropriate for generating general solutions with a performance comparable to that of controller w^1.

The ENS^3-algorithm is of course capable of generating networks for classical network problems - usually solved by feedforward networks. This was reported in [4]. In terms of required computation time (which is large for evolutionary algorithms) ENS^3 can not compete with learning algorithms like backpropagation. Instead, it is used mainly for the development of network structures, optimizing parameters as well, and it has the advantage of producing unconventional topological solutions which may be worthwhile to study in their own right.

The algorithm still can be optimized. For instance the evaluation operator in the variation-evaluation-selection cycle may be substituted by an evaluation-learning cycle, if an appropriate learning procedure is at hand. This is done, for example, in the EPNet-approach in [12] for the case of feedforward networks.

For recurrent networks, and using a behavior based approach to neurocontrollers, there is no universal learning rule to apply. Using only the internal states of a neural network, we are trying to optimize a given recurrent network structure by using ideas outlined in [3]. Furthermore, equivalents to other additional features of evolutionary algorithms - like e.g. crossing over - are not yet implemented in the ENS^3-algorithm.

References

[1] Albrecht, R. F., Reeves, C. R., and Steele, N. C. (eds.) (1993), *Artificial Neural Nets and Genetic Algorithms*, Proceedings of the International Conference in Innsbruck, Austria, 1993, Springer-Verlag, Wien.

[2] Anderson, C. W. and Miller W. T. (1990), Challenging Control Problems, in: W. T. Miller, R. S. Sutton, and P. J. Werbos (eds.), *Neural Networks for Control*, MIT Press, Cambridge.

[3] Der, R., Steinmetz, U., and Pasemann, F. (1999), Homeokinesis - A new principle to back up evolution with learning, in: Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA'99), Vienna, 17-19 February 1999.

[4] Dieckmann, U. (1995), Coevolution as an autonomous learning strategy for neuromodules, in: Herrmann, H., Pöppel, E., and Wolf, D. (eds.), *Supercomputing in Brain Research - From Tomography to Neural Networks*, World Scientific, Singapore, pp. 331-347.

[5] Geva, S., and Sitte, J. (1993), A cartpole experiment benchmark for trainable controllers, *IEEE Control Systems Magazin*, **13**, 40-51.

[6] Pasemann, F. (1998), 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.

[8] Schaffer, J. D., Whitley, D., and Eshelman, L. J. (1992), Combination of genetic algorithms and neural networks: A survey of the state of the art, in: *Proceedings International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)*, Los Alamitos, CA, IEEE Computer Society Press.

[9] Suykens, J. A. K., Vandewalle, J. P. L., and De Moor, B. L. R. (1996), *Artificial Neural Networks for Modelling and Control of Non-Linear Systems*, Dordrecht: Kluver Academic Publishers.

[10] Suykens, J. A. K., De Moor, B. L. R., and Vandewalle, J. P. L. (1994), *Static and dynamic stabilizing neural controllers, applicable to transition between equilibrium points*, *Neural Networks*, **7**, 819–831.

[11] Yao, X. (1993), A review of evolutionary artificial neural networks, *International Journal of Intelligent Systems*, **8**, 539 - 567.

[12] Yao, X., and Y. Liu (1997), A new evolutionary system for evolving artificial neural networks, *IEEE Transactions on Neural Networks*, **8**, 694 - 713.