Balancing Rotators with Evolved Neurocontrollers

Frank Pasemann and Ulf Dieckmann*
Max-Planck-Institute for Mathematics in the Sciences
D-04103 Leipzig, Germany

February 24, 2006

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: Stabilize a rotating pendulum - that is mounted on a cart - in an upright position.

MPI-MIS Preprint 37/1997
*International Institute for Applied Systems Analysis, ADN, A-2361 Laxenburg, Austria

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], [7], [8]). 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. 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. It develops network topology and parameters like weights and bias terms simultaneously. In contrast to genetic algorithms it does not quantize network parameters and it is not only used for optimizing a specific architecture. 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 in the sense of "brains" for autonomous robots or software agents. 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 [5], [6] we reported on first tests of the algorithm, applying it to the pole-balancing problem that usually serves as a benchmark problem for trainable controllers [4]. Although the inverted pendulum is one of the simplest inherently unstable systems, 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 down is therefore a more challenging nonlinear control problem [2]. In section 3 we will show that it is easily solved by evolved neural network solutions.

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.

3 Evolved neurocontrollers

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 $g = 9.81$ denotes gravitational acceleration, $m_c = 1.0$kg and $m = 0.1$kg mass of cart and pendulum, respectively, $l = 0.5$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 $\Delta = 0.01s$.

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 fitness function f for the evaluation of an individual network takes into account costs for each neuron and for each connection (to select for parsimonious network architectures), the pendulum's integrated deviation from the upright position, and the cart's integrated deviation from the zero position. Furthermore, the applied force integrated over the last 20 seconds of each trial can be added as a diminishing contribution to the fitness function. This will optimize the applied force to balance the pendulum by minimizing oscillations of the cart.

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

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

3.1 A t-class controller

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: Its architecture is shown in figure 1a and its weights are given as follows:

$$w = \begin{pmatrix} w_5 \\ w_6 \end{pmatrix} = \begin{pmatrix} 0.0 & -7.56 & -20.0 & -11.88 & 0.0 & 0.0 & -9.66 \\ 0.0 & -1.65 & -17.83 & -3.19 & -4.88 & 0.0 & 0.0 \end{pmatrix},$$

where $w_i = (w_{i0}, w_{i1}, \ldots, w_{in})$ denotes the weight vector of neuron i with w_{i0} the bias term of unit i; here we have $n = 6$.

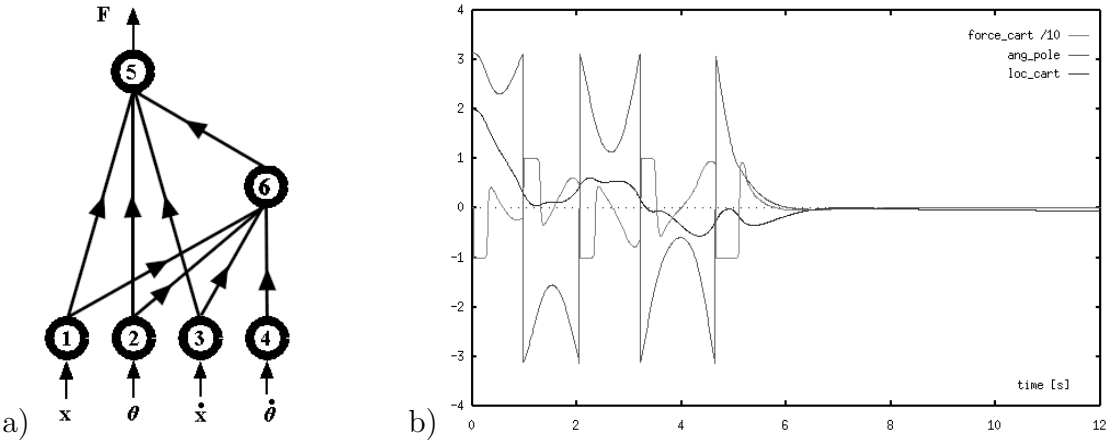

Figure 1: a.) A minimal t-class solution and b.) its effective control: $x(t)$, $\theta(t)$, and $F(t)$ starting from $x_0 = 2.0$ and $\theta_0 = \pi$.

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 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.

3.2 An s-class controller

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 is shown in figure 2a with weights given by

$$\begin{pmatrix} w_5 \\ w_6 \\ w_7 \end{pmatrix} = \begin{pmatrix} 8.2 & 0.0 & -30.46 & -8.38 & 0.0 & 0.0 & 0.0 & -13.34 \\ -6.88 & 21.47 & 12.09 & 12.45 & 0.0 & -12.21 & 0.0 & 28.43 \\ 0.24 & -2.15 & -40.0 & -5.23 & -7.46 & 0.0 & 0.0 & 0.0 \end{pmatrix}.$$

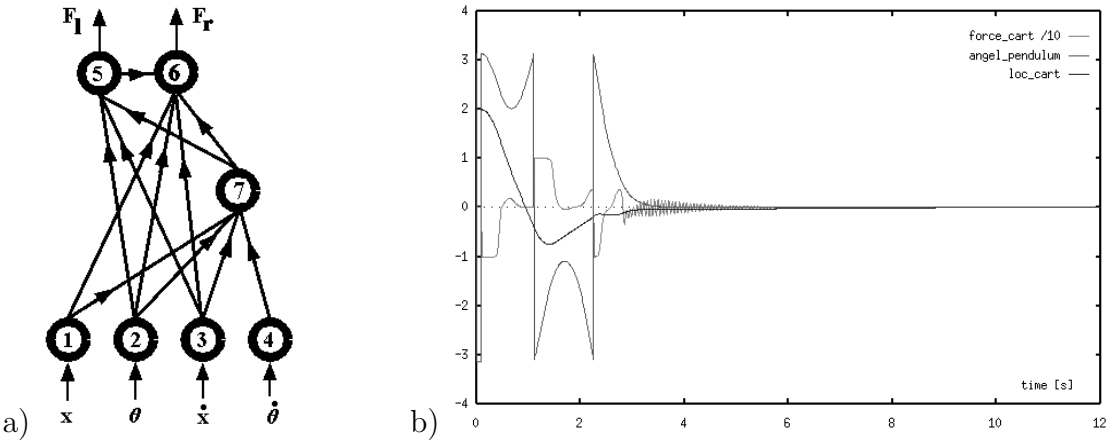

Figure 2: a.) An s-class solution and b.) its effective control: $x(t)$, $\theta(t)$, and $F(t)$ starting from $x_0 = 2.0$ and $\theta_0 = \pi$.

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 2b reveals that it is even faster than the t-class controller. It needs only two swings to get the pendulum in upright position, starting from $x_0 = 2.0$ and $\theta_0 = \pi$. Stabilizing the pendulum and centering the cart is done with an 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.

4 Conclusion

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 are remarkably small in size. They solve the problem very effectively by getting the pendulum in upright position, stabilizing it and centering the cart in less then 10 seconds. Because they have full access to physical phase space variables, they do not need recurrences to compute derivatives. As for the pole-balancing problem discussed in [6], it will be interesting to study the evolved neuromodules for the harder problem where controllers get only information about cart position and angle of the pendulum. Because then velocities have to be computed internally recurrent connectivity is expected to emerge.

That the ENS^3-algorithm is capable of generating networks for classical problems - usually solved by feedforward networks - was reported in [3]. In terms of required computation time (which is large for evolutionary algorithms) ENS^3 can not compete with learning algorithms like backpropagation. Instead, it has the advantage of producing unconventional solutions which are not strictly layered, and 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.

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, *Neural Networks for Control*, MIT Press, Cambridge.

[3] 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.

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

[5] Pasemann, F. and Dieckmann, U. (1997), Evolved neurocontrollers for pole-balancing, in: J. Mira, R. Moreno-Diaz, J. Cabestany (Eds.), Biological and Artificial Computation: From Neuroscience to Technology, Proceedings IWANN'97, Lanzarote, Canary Islands, Spain, June 1997, Springer Verlag, Berlin, pp. 1279 - 1287.

[6] Pasemann, F. (1997), Pole-balancing with different evolved neurocontrollers, in: Gerstner, W., Germond, A., Hasler, M., and Nicoud, J.-D. (eds.), *Artificial Neural Networks - ICANN'97*, October 7-10, Lausanne, Switzerland, Proceedings, LNCS 1327, Springer Verlag, Berlin, pp. 823 - 829.

[7] 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.

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