Evolved Neurocontrollers for Pole-Balancing *

Frank Pasemann
Research Center Jülich, IBI 1
D-52425 Jülich, Germany
email: f.pasemann@kfa-juelich.de

Ulf Dieckmann
International Institute for Applied Systems Analysis, ADN
A-2361 Laxenburg, Austria
email: u.dieckman@iiasa.ac.at

Abstract

An evolutionary algorithm for the development of neural networks with arbitrary connectivity is presented. The algorithm is not based on genetic algorithms, but is inspired by a biological theory of coevolving species. It sets no constraints on the number of neurons and the architecture of a network, and develops network topology and parameters like weights and bias terms simultaneously. Designed for generating neuromodules acting in embedded systems like autonomous agents, it can be used also for the evolution of neural networks solving nonlinear control problems. Here we report on a first test, where the algorithm is applied to a standard control problem: the balancing of an inverted pendulum.

*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.

1 Introduction

The combined application of neural network techniques and genetic algorithms turned out to be a very effective tool for solving an interesting class of problems (for a review see e.g. [10], [5], [14], [1]), especially in situations where there is no good guess for an appropriate network architecture or where recurrent dynamic networks should be used for tasks like generation of temporal sequences, recognition, storage and reproduction of temporal patterns, or control problems which require memory to compute derivatives or integrals.

The algorithm introduced in section 2 is inspired by a biological theory of coevolution and is not based on genetic algorithms. It uses standard additive neuron models 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. Using a behavior based approach to neural systems, the algorithm originally was designed to study the appearence of complex dynamics and the corresponding structure-function-relationship in artificial sensomotoric systems for autonomous robots or software agents. For the solution of extended problems (more complex environments or sensor-motor 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 then genetic algorithms.

Here we report on a first test of the algorithm, applying it to the pole-balancing problem. The inverted pendulum is one of the simplest inherently unstable systems and represents a wide class of control problems requiring avoidance of costly or harmful conditions. Usually it serves as a benchmark problem for trainable controllers [9]. The goal here is to evolve neural controllers able to balance an inverted pendulum mounted on a cart and centering the cart simultaneously. This is decribed in section 3. Besides conventional control techniques, there have been many successfull applications of neural networks to this problem [2], [4], [7], [11], [6], [13], which can be compared with solutions obtained by the presented evolutionary algorithm. Using continuous neurons for the controllers, different from many other results, our approach does not make use of quantization, neither of the physical phase space variables nor of internal parameters, like weitghts and bias terms, and output values. Section 4 gives a discussion of the results.

2 The evolutionary algorithm

We first have to decide which type of neurons to use for the network. We prefer to have the same type of neurons for output and internal units; the input units may be used as buffers as in feedforward networks. Then the number of input and output units is choosen according to the definition of the problem in terms of incoming sensor and outgoing motor 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; but no backward connections to the input units are allowed.

To evolve the desired neuromodule we consider a population $p(t)$ of $n(t)$ neuromodules undergoing a variation-evaluation-selection loop. The variation part of the variation-evaluation-selection loop then allows for the insertion and deletion of neurons and connections. After evaluating the performance of the networks in the population with respect to the problem considered, a new population is generated from the old population. The number of network copies passed from the old to the new population depends on the network's performance. In consequence of this selection process the average performance of the population will either stay the same or increase. Thus, after repeated passes through the variation-evaluation-selection loop networks have build up in the population that solve the considered problem.

We now outline the three major steps of the variation-evaluation-selection loop.

1. *Variation*
   The variation operators include insertion and deletion of neurons, insertion and deletion of connections, and alterations of bias and weight terms. The associated operators acting on the ith member of the population p are denoted by N_i^+, N_i^-, C_i^+, C_i^-, N_i^* and C_i^* respectively. A variation pass is then described by

$$V: \quad p(t) \mapsto \prod_{i=1}^{n(t)} C_i^* C_i^+ C_i^- N_i^* N_i^+ N_i^- \; p(t) \quad .$$

   The operators $N_i^{*,+,-}$, $C_i^{*,+,-}$ are of stochastic character. The chance that they will execute their respective action is determined by fixed per-neuron and per-connection probabilities. In a more complex version the variation operator V may also induce the exchange of entire subnetworks between members of the population p.

2. *Evaluation*

The evaluation operator

$$E: \quad p(t) \mapsto (p(t), e(t))$$

is problem-specific. In its simplest form the performances $e_i(t)$ of the $n(t)$ networks in the population $p(t)$ are mutually independent. As an example, for a classification task the performance of each member of the population could be based on an error function [8], like those utilized by the backpropagation learning algorithm. The evaluation operator usually will be deterministic; more sophisticated versions may also account for network size and past performance. Furthermore, interactions between members of the population can be defined via an artificial sensomotoric loop opening up the potential for coevolutionary dynamics.

3. *Selection*
   Differential survival of the varied members of the population is defined by the selection operator. Possible definitions range from (a) probabilistic survival according to evaluation results to (b) winner-takes-all selection. The selection operator is given by

   $$S: \quad (p(t), e(t)) \mapsto \prod_{i=1}^{n(t)} R_i^{\nu(e_i(t))}\, p(t) \quad .$$

   Here R_i is the reproduction operator for the ith network in the population p. Consequently, $\nu(e_i(t))$ copies of each such network are passed to the new population. In case (a), applied in the following, these integer numbers ν are stochastic variables drawn e.g. from Poisson distributions with mean values larger than 1 if $e_i(t) > \sum_{i=1}^{n(t)} e_i(t)/n(t)$ and mean values smaller than 1 for the other networks in the population p. Case (b) is defined by $\nu(e_i(t)) = n(t)$ if $e_i(t) = \max_i e_i(t)$ and $\nu(e_i(t)) = 0$ otherwise.

The evolution of the population p is then generated by repeated application of the mapping

$$p(t) \mapsto SEV\, p(t)$$

on the initial population p(0).

3 Evolved pole-balancing controllers

The pole-balancing task has three simultaneous objectives: balancing an inverted pendulum, which is mounted on a cart moving in a one-dimensional interval, avoiding the interval boundaries, and centering the cart. We use the standard benchmark values described for instance in [3], i.e. the cart is bound to move in the interval $-2.4 < x < 2.4$ $[m]$, the angle is allowed to vary in the interval $-12 < \theta < 12$ $[^\circ]$. Since we will use neurons with sigmoidal transfer function the force applied to the cart varies continuously between $-10 < F < 10$ $[N]$. We also use additional damping terms according to [4]. The equations for the physical system being controlled is thus given by

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

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

where g denotes gravitational acceleration, $m_c = 1.0$ kg and $m = 0.1$ kg mass of cart and pole, respectively, $l = 0.5$ m half of pole length, $\mu_c = 5 \cdot 10^{-4}$ and $\mu_p = 2 \cdot 10^{-6}$ friction coefficient of cart and pole, respectively, and F denotes the force applied to the cart. The cart-pole-dynamics is computed by using Euler discretization with time step $\Delta = 0.01$ s.

For the neural controller we will use the following standard additive neuron model with sigmoidal transfer function $\sigma = tanh$, i.e.

$$a_i := \sum_{j=1}^{n} w_{ij}\sigma(a_j) + \theta_i$$

A failure signal is given if $|x| > 2.4$ or $|\theta| > 12°$ or balancing time t exceeds a given time t_{max}. The fitness function f for the evaluation of an individual modul takes into account costs for each neuron and for each connection (to obtain kind of minimal networks), and the balancing time until failure. Furthermore, the applied force integrated over the balancing time can enter the fitness function. This will optimize the applied force to balance the pole, i.e. in general this will minimize oscillations of the cart. Thus, the fitness function for an evaluated module has the general form

$$f := P - cost_n \cdot N_n - cost_s \cdot N_s - cost_F \cdot I_F$$

where P denotes the output performance of a module given by

$$P := \sum_{i=1}^{n} (\frac{1}{2}\,(1 - \frac{15 \cdot |\theta(i)|}{\pi}) + (\frac{1}{2}\,(1 - \frac{|x(i)|}{2.4})) \cdot \Delta \quad ,$$

with n the maximal number of iterations; i.e. the maximal balancing time is $t_{max} = n \cdot \Delta$. The constants $cost_n$, $cost_s$, and $cost_F$ describe the costs of a neuron, a synapse, and the applied force, respectively; N_n and N_s denote the number of neurons and of synapses in the module, and the integrated force term I_F is given by

$$I_F = \frac{1}{t_{max}} \sum_{i=1}^{n} |F(i)| \cdot \Delta\ .$$

It turned out, that simulations with populations of 20 - 30 individuals are optimal. During intermediate states of the evolutionary process the fittest modules may become quite large - more than 40 neurons and 100 synapses - and network size and architecture are often varied. Finally there appear smaller modules with equally good or better performance.

3.1 4-input-modules

As sensor signals we first choose the full set of physical statet variables x, θ, $\dot{x}$, $\dot{\theta}$ of the cart-pole-system. This means that we have four input units receiving here the input signals:

$$in_1 := \frac{x}{2.4} \ , \ in_2 := \frac{15 \cdot \theta}{\pi} \ , \ in_3 := \frac{\dot{x}}{2.4} \ , \ in_4 := \frac{15 \cdot \dot{\theta}}{\pi} \ . \tag{1}$$

The output unit provides the force applied to the cart by

$$F = 10 \cdot \sigma(a_5(t)) \ .$$

For initial conditions confined to the benchmark domain, it is well known (see e.g. [12]) that there exist neural network solutions using only the output unit and no internal neurons. Different from the classical networks, which used binary output units $(-1, 1)$ providing a pulsed force to the cart (bang-bang control), here a continuous force is applied. The following is a test, if the evolutionary algorithm is able to generate such minimal solutions. In fact, besides a whole family of larger modules, it came up with these minimal solutions. One of them is shown in figure 1a. Its weight vector is given by

$$w_5 = (\theta_5, w_{51}, \ldots, w_{54}) = (0.031, 14.68, 13.88, 20.00, 3.41) \ .$$

We test this configuration on 40 x 40 benchmark initial conditions (x, θ, $\dot{x} = \dot{\theta} = 0.0$) represented by squares in figure 1b. Outer squares represent initial conditions which will already produce a failure signal. We observe, that the module avoids the ends of the interval very effectively; furthermore it centers the cart, and balances a long time ($>$ 120 s). Although the module was evolved with initial conditions x and θ inside the benchmark domain, we observe, that it performs well also on initial condition far outside this domain. Other modules which evolved in simulations, using for instance one hidden neuron, performed equally good. Some of them utilized also internal oscillators, which came into action only if the physical system reached critical phase space domains. If there was no optimizing condition for the applied force ($cost_F = 0$), almost all solutions solved the balancing problem with a continuously oscillating cart.

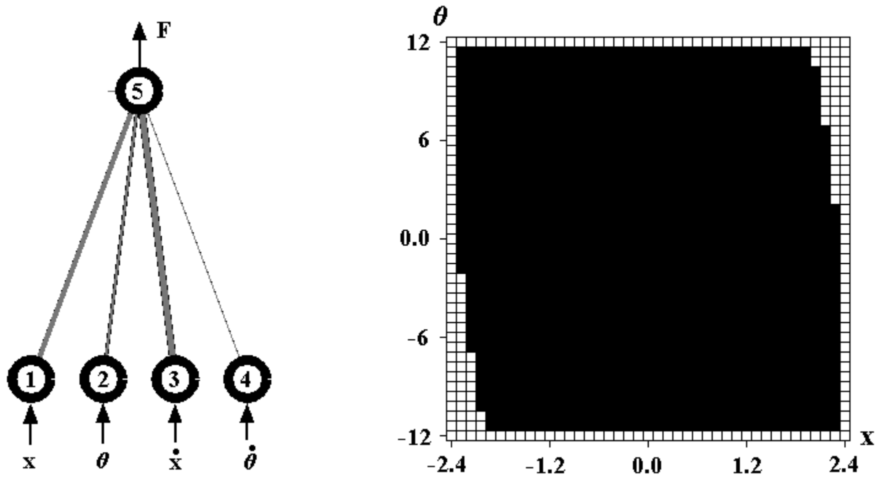

Figure 1: a.) A minimal solution with four inputs and b.) its performance on benchmark initial conditions. Balancing time $t > 120$ s: black and $t < 30$ s: white.

3.2 2-input-modules

Reducing the inputs to the control module to only the cart position and the pole angle makes the problem for the controller more sophisticated. It now has to compute the derivatives $\dot{x}$ and $\dot{\theta}$, and therefore modules with recurrent connections have to be expected. As inputs we choose

$$in_1 := \frac{x}{2.4} \quad , \quad in_2 := \frac{15 \cdot \theta}{\pi} \; . \tag{2}$$

The simplest kind of evolved modules used in fact only the output neuron with a self-connection and no hidden units, as shown in figure 2a. Although these solutions do balancing and avoiding on benchmark initial conditions quite well (compare figure 2b), they do not center the cart. Instead, they let the cart oscillate around the zero position with an apparently constant amplitude, without hitting the walls. The amplitude is given by the initial cart position. The solution displayed in figure 2 has the weight vector

$$w_3 = (\theta_3, w_{31}, \ldots, w_{33}) = (0.0, -1.23, 16.22, -0.72) \; .$$

Again, also larger modules evolved with equally good performance, but none of them suppressed the cart oscillations.

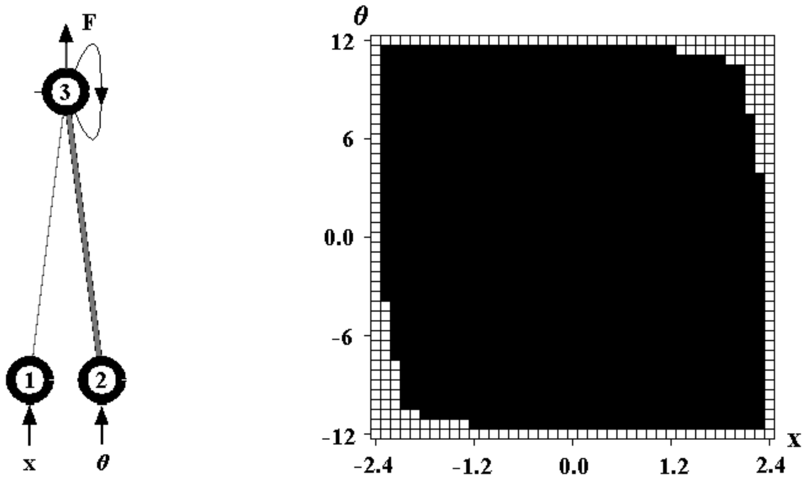

Figure 2: a.) A minimal solution with two inputs and b.) its performance on benchmark initial conditions. Balancing time $t > 120$ s: black and $t < 30$ s: white.

4 Discussion

We have demonstrated that the evolutionary algorithm presented here can be applied successfully to the pole balancing problem. The evolved solutions are remarkable small when compared with solutions obtained by classical control techniques or pure feedforward networks. We also observe a strong effect of the "survival conditions" on the topology of evolved networks: If there is no condition to minimize the applied force for balancing, solutions tend to use internal oscillators realized for instance by inhibitory self-connections. We have also observed solutions making use of switched oscillators, which come into action only if the physical system enters critical phase space domains. Furthermore, the evolved controllers were quite robust in the sence, that a moderate noise on the input signals did not effect their performance.

Since we have not found yet 2-input modules which suppress the oscillations of the cart, we evolved controllers, which only had to center the cart starting from any position in the benchmark interval. With two inputs $(x, \dot{x})$ simple solutions evolved, which brought the cart into the central position without any swing around zero. Using only the cart position as input, the cart still oscillated around zero, but now with a damped oscillation, bringing the cart finally at rest at zero. But these solutions did not survive, when implemented into the extended problem of balancing the pole simultaneously.

We used also different setups to study corresponding solutions. For example we used neurons with standard sigmoidal transfer function $\sigma(x) =$

$(1+e^{-x})^{-1}$. Then two output units are needed, providing a force given e.g. by $F = 10 \cdot (\sigma(a_5) - \sigma(a_6))$ for a 4-input module. The evolutionary algorithm provided a couple of interesting solutions, among others, a minimal 4-input solution with only four connections to one output unit (unit 5), and a second constant output (unit 6). This solution is of course comparable with the above solution for the *tanh* transfer function.

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 of course the case for classical problems solved by feedforward networks. That the evolutionary algorithm introduced here is capable to generate also such solutions was reported in [8], although, due to the extensive computation time, it can not compete with learning algorithms like backpropagation. Instead it has the advantage, that it produces unconventional not strictly layered solutions which may be worth-while to study.

Besides the first test presented here, further simulations on more difficult problems, like for instance balancing a rotator, indicate, that the evolutionary algorithm may be effectively applied to many nonlinear control problems.

References

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

[2] Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, **9**, 31 - 37.

[3] Anderson, C. W. and Miller W. T. (1990). Challinging Control Problems. In W. T. Miller, R. S. Sutton, and P. J. Werbos, *Neural Networks for Control*, MIT Press.

[4] Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that solve difficult learning control problems. *IEEE Transactions on Systems, Man, Cybernetics*, **13**, 834 - 846.

[5] Branke, J. (1995). Evolutionary algorithms for neural network design and training. In *Proceedings 1st Nordic Workshop on Genetic Algorithms and its Applications*, Vaasa, Finland.

[6] Bapi, R. S., D'Cruz, B., and Bugmann, G. (1996) Neuro-resistive grid approach to trainable controllers: A pole balancing example, submitted to *Neural Computing and Applications.*

[7] Dasgupta, D., and McGregor, D. R. (1993). Evolving neurocontrollers for pole balancing. In S. Gielen and B. Kappen (eds.), *ICANN'93 Proceedings of the International Conference on Artificial Neural Networks*, Amsterdam 13.-16. Sept. 1993. Berlin: Springer-Verlag, 1993, pp. 834-837.

[8] 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*, Singapore: World Scientific, (pp. 331–347).

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

[10] 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.

[11] Selfridge, O. G., Sutton, R. S., and Barto, A. G. (1985). Training and tracking in robotics. In *Proceedings International Joint Conference on Artificial Intelligence (IJCAI-85)*, Los Angeles, CA, pp: 670 - 672.

[12] Widrow, B. (1987), The original adaptive neural net broom-balancer. *Proc. IEEE Intern. Symp. Circuits and Systems*, 351 - 357.

[13] Wieland, A. P. (1991). Evolving neural network controllers for unstable systems. In: *International Joint Conference on Neural Networks*, Seattle, USA, July1991. Proccedings Vol.**2**. Seattle: IEEE Service Center, 1991.

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