Architecture optimisation of a network of dynamic neurons using the A*-algorithm

Andrzej Obuchowicz
Dept. of Robotics and Software Engineering, Technical University of Zielona Góra
ul. Podgórna 50, 65-246 Zielona Góra, Poland
Phone: +48 68 3254831 ext. 422, Fax: +48 68 3254615
email: A.Obuchowicz@irio.pz.zgora.pl.

ABSTRACT: The modelling of the dynamic non-linear system is the aim of this work. A neural model is constructed with dynamic artificial neurons, which contain inner feedbacks. Such a model consists of an adder module, a linear dynamic system and a non-linear activation function. Emphasis is putted on the searching of the optimal, in the sense of generalisation capability, network architecture. On the basis of a construction of a graph of network architectures and an evaluation values which are assigned to them, an heuristic search algorithm can be installed on this graph. The application of the A*-algorithm ensures, in theory (Doering *et al*., 1997), both the optimality of the solution and the optimality of the search. An illustrative example is introduced.

KEYWORDS: dynamic system modelling, network of dynamic neurons, network architecture optimisation generalisation capability, heuristic search.

INTRODUCTION

The application of Artificial Neural Networks (ANN) in modelling and identification of the dynamic processes has been intensively studied for the last two decades (cf. Narendra and Parthasarathy, 1990; Korbicz *et al.*, 1998). Attractiveness of ANN's follows from the fact that they are useful when there are no mathematical models of investigated system, hence, analytical models and parameter-identification algorithms cannot be applied. As opposed to a lot of ANN effective applications, e.g. in the pattern recognition or in the approximation of the non-linear function, the application of ANN's in modelling requires taking into consideration the dynamics of the investigated processes. In order to apply the ANN with a Multilayer Feedforward Perceptron (MFP) architecture and the Back-Propagation (BP) training algorithm to dynamic system identification, one should artificially introduce delay elements (cf. Korbicz, 1995), because the MFP is a static network (Hertz *et al*., 1991). Hence the possibilities of its application in dynamic problems are limited. Recurrent neural networks are characterised by considerably better properties assessing from the point of view of their application in control theory (Tsoi and Back, 1994; Draye *et al*., 1996). Unfortunately, practical realisations of such a network structure are very limited, mainly due to their instability and a very slow convergence of the training process. The Elman recurrent network has less general character but better characteristics of practical applications (Elman, 1990). It is worth noting that the standard recurrent neural networks are built using the static McCulloch-Pitts neuron model.

One of the most interesting solutions of dynamic system modelling problem is the application of the Dynamic Neural Model (DNM) (Ayoubi, 1994; Patan and Korbicz, 1996). Such a neuron model consists of an adder module, a linear dynamic system - Infinite Impulse Response (IIR) filter, and a non-linear activation module. The relatively complex DNM allows one for building an effective feedforward multilayer Network of Dynamic Neurons (NDN). The NDN can have the same architecture as the MFP. The calculated output error is propagated back to the input layer through hidden layers containing dynamic filters, similarly as in the standard BP algorithm. As a result, the Extended Dynamic Back-Propagation algorithm may be defined (Patan and Korbicz, 1996). This algorithm adjust connection weights as well as IIR filter parameters. Unfortunately, the training process of an NDN which has to identify a dynamic system, seems to be an optimisation problem which is intrinsically related to a very rich topology of the sum square-error function

(Korbicz *et al.*, 1998). The EDBP algorithm usually finds one of the local unsatisfactory optima. High performance of a dynamic system neural modelling, which has been trained by an evolutionary algorithm, has been observed by Obuchowicz (1999). There has been implemented the Evolutionary Search with Soft Selection and Forced Direction of Mutation (ESSS-FDM) algorithm (Obuchowicz and Korbicz, 1998).

The choice of an optimal architecture for an ANN with the MFP structure, by which a given problem should be solved, is an important prerequisite problem for research workers. The MFP architecture influences the rate of training procedure, network processing rate and quality (Hush and Horne, 1993, Korbicz *et al*., 1994). The crucial tradeoff one has to make is between the learning capability of the MFP and fluctuations due to finite sample size. If the architecture of the MFP is too small, network might not be able to approximate well enough the functional relationship between the input and target output. If the number of free parameters is too great (compared to the number of training samples), the realised network function will depend too much on the actual realisation of the training set. The number of neurons in input and output layers is usually determined by the dimensions of input **u** and output **y** vectors. The problem is to set an appropriate number of hidden layers, the number of neurons in each of these layers and the number of connections.

There are very rich bibliography items and various methods to solve this problem (c.f. Doering *et al.*, 1997; Fahlman and Liebere, 1990; Obuchowicz and Politowicz, 1997; Obuchowicz and Patan, 1998). Recently, a variety of architecture optimisation algorithms have been proposed. They can be divided into three classes (Doering *et al.*, 1997) :

1) *discrete optimisation methods*: each network architecture is assigned an evaluation value and the network architecture space is searched by a given algorithm, in particular, genetic algorithms seem to have gained a strong attraction within this context.
2) *bottom-up approaches*: starting with a relatively small architecture, these procedures increase the number of hidden neurons and thus increase the power of the growing network;
3) *top-down approaches*: these procedures start with a network architecture that is supposed to be sufficiently complex to model the relation between input and output variables, after training they try to reduce the number of hidden neurons as much as possible;

The first class of methods proves to be the most flexible approach, though computationally expensive, complexity of all known algorithms is exponential. Several bottom-up methods have been reported to train even hard problems with a reasonable computational effort. The resulting network architectures can hardly be proven to be optimal. But, a further criticism concerns the insertion of hidden neurons as long as elements of the training set are misclassified. Thus the resulting networks posses a poor generalisation performance and are disqualified for many applications. The top-down approaches inherently assume knowledge of a sufficiently complex network architecture that can always be provided for finite size training samples. Because the algorithms presented up to now can only handle special cases of redundancy reduction in a network architecture, they are likely to result in a network that still oversized.

One of the most interesting approaches, proposed by Doering *et al.* (1997), belongs to the first class, where the crucial point certainly is the efficient use of information already gained during training a sequence of network architectures. The A*-algorithm is applied. It is known that it uses heuristic information in a optimal way and thus is superior to all other algorithms working with the same heuristic information, i.e., it finds the optimal architecture by exploring the smallest possible subset of the search space

The significance of network architecture optimisation increases when the NDN is taken under considerations. The number of free network parameters rapidly increases when one substitutes standard MFP neurons by the DNM units. Thus, there is some quality difference between architecture allocation of the MFP and the NDN. Apart from setting an appropriate number of hidden layers and the number of neurons in each of these layers, the dynamic order of each particular neuron have to be established in the NDN.

The aim of this work is to construct the optimal NDN to model a given dynamic non-linear system. In order to solve our problem the A*-algorithm is implemented.

In the following, the NDN is introduced. Next, the problem statement is given. Finally, the A*-algorithm approach to the NDN architecture allocation is discussed and illustrated.

NETWORK OF DYNAMIC NEURONS

DYNAMIC NEURAL MODEL

In this Section, the extended structure of the neuron model - DNM - proposed by Ayoubi (1994) is considered. The dynamics is introduced to a neuron in such a way that the neuron activity depends on its internal states. It is done by adding a linear dynamic system, IIR filter, to the neuron structure. In this way, each neuron in the dynamic network reproduces past signal value using two signals: the inputs $u_p(k)$ for p=1,2,...,P and the output $y(k)$. In the Fig. 1, the structure of the DNM is shown.

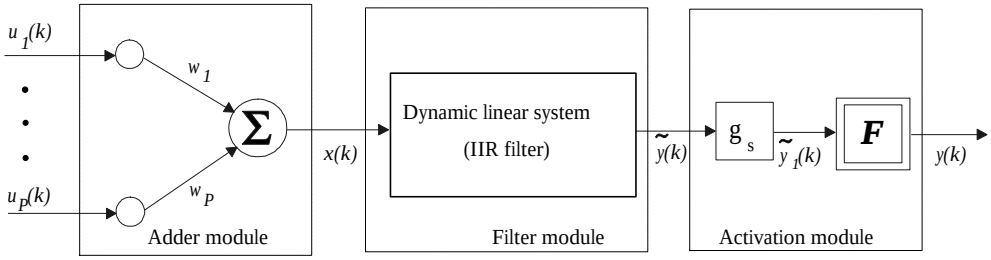

Figure 1: General structure of the dynamic neuron model with P inputs.

The dynamic structure of the DNM, as shown in Fig. 1, consists of three submodules: the adder, the filter and the activation module. In the adder module the weighted sum $x(k)$ of inputs is calculated according to the formula:

$$x(k) = \mathbf{w}^T \mathbf{u}(k) = \sum_{p=1}^{P} w_p u_p(k), \tag{1}$$

where $\mathbf{w} = \left[w_1\ w_2\ \ldots\ w_P\right]^T$ denotes the input weights vector, $\mathbf{u}(k) = \left[u_1(k)\ u_2(k)\ \ldots\ u_P(k)\right]^T$ is the input vector.

Then the calculated sum $x(k)$ is passed to the filter module. Here the filters under consideration are the linear dynamic systems of different orders, viz. the first, second and third order. The general structure of the n-th order IIR filter is shown in Fig. 2. This filter consists of delay elements (denoted by z^{-1}) and feedback and feedforward paths weighted by the synaptic vector weights $\mathbf{a} = \left[a_1\ \ldots\ a_n\right]^T$ and $\mathbf{b} = \left[b_0\ b_1\ \ldots\ b_n\right]^T$, respectively.

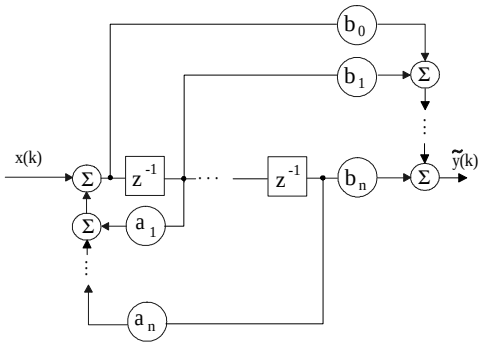

Figure 2: Block scheme n-th order IIR filter.

The behaviour of this module can be described by the n-th order difference equation given by:

$$\tilde{y}(k) = -a_1 \tilde{y}(k-1) - \ldots - a_n \tilde{y}(k-n) + b_0 x(k) + b_1 x(k-1) + \ldots + b_n x(k-n), \tag{2}$$

where $x(k)$ and $\tilde{y}(k)$ are the filter input and output, respectively, and k is the discrete-time.

Then, $\tilde{y}(k)$ is the input for the next module - the activation module. Finally, neuron output, which is the output of the activation module, can be described by:

$$y(k) = F\left(\tilde{y}_1(k)\right) = F\left(g_s \tilde{y}(k)\right), \tag{3}$$

where $F(\bullet)$ is some non-linear activation function that produces the neuron output $y(k)$, and g_s is the slope parameter of the activation function. In a dynamic neuron, the slope parameter can change its value.
Let us define the activation function of the DNM unit as a the hyperbolic tangent function given by :

$$F\left(\tilde{y}_1(k)\right) = \tanh\left[g_s \tilde{y}(k)\right] = \frac{e^{[g_s \tilde{y}(k)]} - e^{-[g_s \tilde{y}(k)]}}{e^{[g_s \tilde{y}(k)]} + e^{-[g_s \tilde{y}(k)]}}. \tag{4}$$

In limit, when $g_s \to \infty$, hyperbolic tangent function tends to become the signum function.

NETWORK OF DYNAMIC NEURONS

Considering the described above dynamic neuron model, one can design more complex structure - a neural network. Using the well-known MFP structure with DNM units as nodes a network of dynamic neurons (Fig. 3) is defined. In comparison with the Elman network and recurrent network with outside feedbacks, which are most popular neural networks applied to the dynamic system modelling, the proposed NDN does not require past values of the input vector (process measurement). Instead, it processes the modelled system measurements at current instant *k*. This reduces the dimension of the network input space.

In Fig. 3, *M*-layer architecture of the dynamic network is presented. The input elements are denoted out as a squares. They preliminary transform the data and pass them on to the first neural layer. The neural processing elements - DNM units are denoted out as circles. The DNM units are organised in layers, where the last *M*-layer is called the output layer and other layers from *1* to *M-1* are called the hidden layers. The inputs flow from layer to layer only in one direction - ahead.

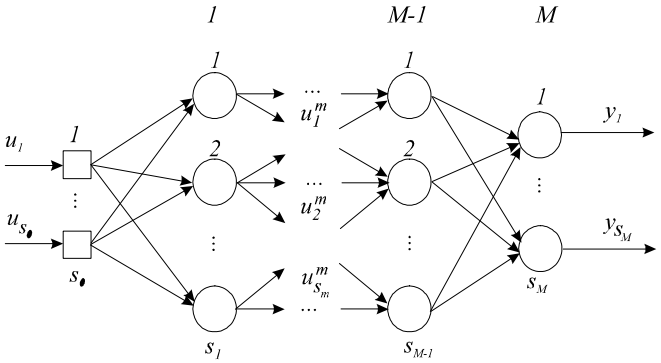

Figure 3: Schema of the NDN architecture.

The most commonly applied neuron network models are the MFP and the Radial Basis Function (RBF) networks. Both networks are proved to be universal approximators of static non-linearities (Cybenko, 1989; Hecht-Nelsen, 1990; Hornik *et al.*, 1989). The proof that the NDN can be universal identifier may be based on the Leontaritis and Billings theorem (1985). They prove that under some assumptions, any non-linear, discrete and time invariant system can be always represented by the following simplified version of a NARMAX model (Non-linear Auto Regressive Moving Average with eXogenous inputs) :

$$y(k) = f\left(u(k), u(k-1), \ldots, u(k-m), y\left(k\text{-}1\right), \ldots, y\left(k\text{-}n\right)\right), \tag{5}$$

where $y(k-i)$ and $u(k-i)$ are outputs and input past values at instant $(k-i)$ for $i = 1, \ldots, m, \ldots, n$.

Summing up we may provide a formal definition of the NDN. The NDN presents an ordered pair $(NA, \mathbf{v})$. *NA* denotes the network architecture

$$NA = \left(\left(V_i \middle| i = 0, \ldots, M \right), \left(o^{i}_{j} \middle| i = 1, \ldots, M;\ j = 1, \ldots, s_i \right), \Lambda \right). \tag{6}$$

$\left(V_i \middle| i = 0, \ldots, M \right)$ is a family of $M+1$ sets of DNM units, called layers, including two disjunctive non-empty sets V_0, V_M that define the input (non-processing) and output (processing) units, respectively. $\left(o^{i}_{j} \middle| i = 1, \ldots, M;\ j = 1, \ldots, s_i \right)$ is set of natural numbers, o^{i}_{j} denotes the IIR dynamic order of the *j*-th DNM unit from the *i*-th layer, o^{i}_{j}. $\Lambda = \bigcup_{i=0}^{M-1} V_i \times V_{i+1}$ is a set of edges that define the connections between units in the network.

The vector of network parameters **v** can be expressed in the following way :

$$\mathbf{v} = \left(\mathbf{w}, \left(\mathbf{a}^{i}_{j}, \mathbf{b}^{i}_{j}, g^{i}_{sj} \middle| i = 1, \ldots, M;\ j = 1, \ldots, s_i \right) \right), \tag{7}$$

where the set of weights $\mathbf{w}: \Lambda \to \Re$ assigns a real value to each connection, $\mathbf{a}^{i}_{j}, \mathbf{b}^{i}_{j}, g^{i}_{sj}$ describes feedback and feedforward IIR synaptic vectors and the slope parameter, respectively, of the *j*-th DNM unit from the *i*-th layer.

PROBLEM STATEMENT

Let

$$\mathbf{y}(k) = f\left(\mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-m), \mathbf{y}(k-1), \ldots, \mathbf{y}(k-n) \right) \tag{8}$$

is the response of a non-linear dynamic system $f(\bullet)$ on an input signal $\mathbf{u}(k)$. Let $\Phi = \left\{ \mathbf{u}: K \to \Re^{s_0} \right\}$ is a family of all possible maps (infinitely many) from the set *K* of discrete time moments to the space $\Re^{s_0}$ of input signals. The ultimate goal of the construction of a neural model (with an architecture NA and a set of network parameters **v**)

$$\mathbf{y}_{NA,\mathbf{v}}(k) = f_{NA,\mathbf{v}}\left(\mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-m_{NA}), \mathbf{y}_{NA,\mathbf{v}}(k-1), \ldots, \mathbf{y}_{NA,\mathbf{v}}(k-n_{NA}) \right) \tag{9}$$

of a dynamic system (8) is the minimisation of a cost function $\sup_{\mathbf{u}(k) \in \Phi} J_T\left(\mathbf{y}_{NA,\mathbf{v}}(k), \mathbf{y}(k) | k \in K \right)$. Thus one has to determine the following pair :

$$\left(NA^{opt}, \mathbf{v}^{opt} \right) = \arg\min \left[\sup_{\mathbf{u}(k) \in \Phi} J_T\left(\mathbf{y}_{NA,\mathbf{v}}(k), \mathbf{y}(k) | k \in K \right) \right]. \tag{10}$$

Practically, the solution of the (10) cannot be achieved, because of infinite size of Φ. Hence, in order to estimate the solution two finite subsets $\Phi_L, \Phi_T \subset \Phi : \Phi_L \cap \Phi_T = \varnothing$ are separated. The set Φ_L is used to determine the best vector of parameter of a given network architecture *NA*:

$$\mathbf{v}^{*} = \arg\min_{\mathbf{v} \in V} \left[\max_{\mathbf{u}(k) \in \Phi_L} J_L\left(\mathbf{y}_{NA,\mathbf{v}}(k), \mathbf{y}(k) | k \in K \right) \right], \tag{11}$$

where *V* is a space of network parameters. In general, the cost function for a training process $J_L\left(\mathbf{y}_{NA,\mathbf{v}}(k), \mathbf{y}(k) | k \in K \right)$ may have different definition from the testing cost function $J_T\left(\mathbf{y}_{NA,\mathbf{v}}(k), \mathbf{y}(k) | k \in K \right)$.

The set Φ_T is used to determine the network architecture *NA* that realises the minimal cost within the set of all network architectures $A = \{NA\}$:

$$NA^* = \arg\min_{NA \in A} \left[\max_{\mathbf{u}(k) \in \Phi_T} J_T \left(\mathbf{y}_{NA,\mathbf{v}^*}(k), \mathbf{y}(k) | k \in K \right) \right]. \tag{12}$$

Obviously, the solutions of both tasks, (11) and (12), need not necessary be unique. If several architectures fulfil the latter criterion, the one with minimal free network parameters is defined as optimal.

THE A*-ALGORITHM APPROACH

The A*-algorithm was first proposed by Nilson (1980). Its approach to the architecture optimisation of the MFP has been implemented by Doering *et al.* (1997). Below the A*-approach to the NDN architecture optimisation is described.

The A*-algorithm works on a graph whose arcs are assigned costs. A subset of the set of nodes is the goal set, the elements of which fulfil a given success criterion. The aim of the search performed by A*-algorithm is twofold (Doering *et al.*, 1997):

1) if the solution exists, to find an element of the goal set (admissibility);
2) to find a solution with a minimal search effort in terms of the number of expanded nodes (optimality).

The basic idea of the A*-algorithm is that given any node n_i of the graph the cost of optimal path from this one to the „nearest" goal node n_G^* (in terms of costs) are described by a function $h^*(n_i)$. Of course, neither this optimal path nor its cost are known and the estimation $h(n_i)$ of the cost is used. The $h(n_i)$ evaluates some heuristics known about search problem, then the evaluation function

$$f(n_i) = g(n_i) + h(n_i) \tag{13}$$

where $g(n_i)$ is the cost of the best known path from the start node n_0 to the node n_i, quantifies how „promising" the expantion of the node n_i is.

In order to apply the A*-algorithm to an architecture optimisation of the NDN we have to define:

- the set $G \subset A$ of the goal network architectures,
- an expansion operator $\Gamma(NA)$: $A \to 2^A$ that maps any network architecture $NA \in A$ onto a set of successors (which is a subset of *A*),
- the cost function $g(NA, NA')$ assigned to each expansion operation and
- the heuristic function $h(NA)$.

An evaluation criterion that determines the goal set *G* is defined as follows:

$$G = \left\{ NA \in A \left| \max_{\mathbf{u}(k) \in \Phi_T} J_T \left(\mathbf{y}_{NA,\mathbf{v}^*}(k), \mathbf{y}(k) | k \in K \right) \le \eta_0 \right. \right\} \tag{14}$$

where η_0 is properly chosen nonnegative threshold.

The expansion operator $\Gamma(NA)$ creates the following successors.

1) Varying the number of hidden layers. Assume the NDN architecture *NA* (6). The architecture $NA^{(1)}$ with a inserted hidden layer with s_M DNM units

$$
\begin{aligned}
&NA^{(1)} = \left(\left(V_i^{(1)} \middle| i = 0, \ldots, M+1 \right), \left(o_j^{(1)i} \middle| i = 1, \ldots, M+1;\ j = 1, \ldots, s_i \right), \Lambda^{(1)} \right) \\
&V_i^{(1)} = V_i \quad \text{for } i < M \\
&V_M^{(1)} = V_M \\
&V_{M+1}^{(1)} = V_M \\
&o_j^{(1)i} = o_j^{i} \quad \text{for } i < M,\ j = 1, \ldots, s_i \\
&o_j^{(1)M} = 0 \quad \text{for } j = 1, \ldots, s_M \\
&o_j^{(1)M+1} = o_j^{M} \quad \text{for } j = 1, \ldots, s_{M+1} \left(= s_M \right) \\
&\Lambda^{(1)} = \Lambda \cup \left\{ \left(\upsilon_1^{(1)}, \upsilon_2^{(1)} \right) \middle| \upsilon_1^{(1)} \in V_M^{(1)}, \upsilon_2^{(1)} \in V_{M+1}^{(1)} \right\}
\end{aligned}
\tag{15}
$$

is the successor of *NA*.

2) Varying the number of units in a hidden layer. Assume the architecture *NA* (6) that has at least one hidden layer ($M \geq 2$). Then, all architectures $NA^{(2)}$ with an inserted unit $\upsilon^{(2)}$ in the *m*-th layer for $m = 1, \ldots, M-1$

$$
\begin{aligned}
&NA^{(2)} = \left(\left(V_i^{(2)} \middle| i = 0, \ldots, M \right), \left(o_j^{(2)i} \middle| i = 1, \ldots, M;\ j = 1, \ldots, s_i \right), \Lambda^{(2)} \right) \\
&V_i^{(2)} = V_i \quad \text{for } i \neq m \\
&V_m^{(2)} = V_m \cup \upsilon^{(2)} \\
&o_j^{(2)i} = o_j^{i} \quad \text{for } j = 1, \ldots, s_i \\
&o_{s_m+1}^{(2)m} = 0 \\
&\Lambda^{(2)} = \Lambda \cup \left\{ \left(\upsilon, \upsilon^{(2)} \right) \middle| \upsilon \in V_{m-1}^{(2)} \right\} \cup \left\{ \left(\upsilon^{(2)}, \upsilon \right) \middle| \upsilon \in V_{m+1}^{(2)} \right\}
\end{aligned}
\tag{16}
$$

are successors of *NA*.

3) Varying the IIR order of a DNM unit. Assume the architecture *NA* (6). Then, all architectures $NA^{(3)}$ with an increased order of the IIR filter in the *n*-th DNM unit of the *m*-th layer $m = 1, \ldots, M;\ n = 1, \ldots, s_m$

$$
\begin{aligned}
&NA^{(3)} = \left(\left(V_i \middle| i = 0, \ldots, M \right), \left(o_j^{(2)i} \middle| i = 1, \ldots, M;\ j = 1, \ldots, s_i \right), \Lambda \right) \\
&o_j^{(3)i} = o_j^{i} \quad \text{for } i \neq m,\ j = 1, \ldots, s_i \\
&o_j^{(3)m} = o_j^{m} \quad \text{for } j \neq n \\
&o_n^{(3)m} = o_n^{m} + 1
\end{aligned}
\tag{17}
$$

are successors of *NA*.

Thus, $\Gamma(NA)$ maps an architecture with $M-1$ hidden layers and *N* processing DNM units onto $M+N$ successors. Any NDN architecture $NA \in A$ is accessible from an initial architecture NA_0 without hidden layers, with numbers of the input and output layers specified by the input/output signal dimensions of modelled system and all output DNM units with zero order IIR filters. It can be proved (Doering *et al.*, 1997) that

$$
\forall NA' \in \Gamma(NA)\ \exists \mathbf{v}':\quad J_T\left(\mathbf{y}_{NA', \mathbf{v}'}(k), \mathbf{y}(k) \middle| k \in K \right) \leq J_T\left(\mathbf{y}_{NA, \mathbf{v}^*}(k), \mathbf{y}(k) \middle| k \in K \right) \tag{18}
$$

Each expansion operation $NA' \in \Gamma(NA)$ is assigned a cost vector

$$
g\left(NA, NA' \right) = \begin{bmatrix} \gamma(NA') - \gamma(NA) \\ \delta(NA') - \delta(NA) \end{bmatrix} \tag{19}
$$

where $\gamma(NA) = \sum_{i=1}^{M}\sum_{j=1}^{s_i} o_i^j + \sum_{i=0}^{M-1}\sum_{j=1}^{s_i}\sum_{l=1}^{s_{i+1}} w_{lj}^{(i,i+1)}$ is a number of free network parameters, $w_{lj}^{(i,i+1)}$ is the weight of the connection between the *j*-th unit of the *i*-th layer and the *l*-th unit of the (*i*+1)-st layer, $\delta(NA) = M - 1$ is a number of hidden layers

Each network architecture *NA* can be assigned network parameters $\mathbf{v}^*$ which is solution of (11). Then let

$$h(NA) = \frac{1}{\max\limits_{\mathbf{u}(k)\in\Phi_T} J_T\left(\mathbf{y}_{NA,\mathbf{v}_0^*}(k), \mathbf{y}(k) | k \in K\right)} \left[\begin{array}{c} \max\limits_{\mathbf{u}(k)\in\Phi_T} J_T\left(\mathbf{y}_{NA,\mathbf{v}^*}(k), \mathbf{y}(k) | k \in K\right) \\ 0 \end{array}\right] \tag{20}$$

be the heuristic vector that is associated with the architecture *NA*. $\max\limits_{\mathbf{u}(k)\in\Phi_T} J_T\left(\mathbf{y}_{NA,\mathbf{v}_0^*}(k), \mathbf{y}(k) | k \in K\right)$ denotes the generalisation error of the optimally trained initial network architecture NA_0.

Finally, we have to define the relation $\leq$ in $\Re^2$ (Doering *et al*., 1997) :

$$\forall \mathbf{p}, \mathbf{q} \in \Re^2 \quad \left(\mathbf{p} \dot{\leq} \mathbf{q}\right) \Leftrightarrow \left(p_1 \leq q_1\right) \vee \left(\left(p_1 = q_1\right) \wedge \left(p_2 \leq q_2\right)\right). \tag{21}$$

ILLUSTRATIVE EXAMPLE

As an illustrative example the dynamic system, described by the following difference equation (Narendra and Parthasarathy, 1990):

$$y(k) = f\left(y(k-1), y(k-2), u(k), u(k-1), y(k-2)\right), \tag{22}$$

where the non-linear function $f(\bullet)$ has the form

$$f\left(x_1, x_2, x_3, x_4, x_5\right) = \frac{x_1 x_2 x_3 x_5 \left(x_3 - 1\right) + x_4}{1 + x_2^2 + x_3^2}, \tag{23}$$

is chosen as a system to be modelled.
The set Φ_L consists of only one type of input signal - the white noise. The network squared error

$$J_L\left(y_{NA,\mathbf{v}}(k), y(k)\right) = \left(y_{NA,\mathbf{v}}(k) - y(k)\right)^2 \tag{24}$$

is chosen as a cost function for the *on-line* EDBP training process. The training process ends if $J_L < 0.01$ or the assumed maximum number of iterations (k_{max}=10000) is achieved.
The set Φ_T consists of the following maps:

1) $u(k) = \cos(2\pi k / 360)$; (25)

2) $u(k) = \begin{cases} \sin(2\pi k / 250) & \text{for} \quad k < 250 \\ 0.8\sin(2\pi k / 250) + 0.2\sin(2\pi k / 25) & \text{for} \quad k \geq 250 \end{cases}$; (26)

3) $u(k) = \begin{cases} 0 & \text{for} \quad k < 100 \\ 1 & \text{for} \quad 100 \leq k < 250 \\ \cos(2\pi k / 360) & \text{for} \quad k \geq 250 \end{cases}$; (27)

$$4)\quad u(k)=\begin{cases}0 & \text{for}\quad k<150\\ 1 & \text{for}\quad 150\le k<250\\ 1+\sin(2\pi k/25) & \text{for}\quad k\ge 250\end{cases}; \tag{28}$$

$$5)\quad u(k)=\begin{cases}0 & \text{for}\quad k<120\\ 1 & \text{for}\quad 120\le k<380\\ 0 & \text{for}\quad k\ge 380\end{cases}; \tag{29}$$

$$6)\quad u(k)=\begin{cases}0.25\,1\sin(2\pi k/25) & \text{for}\quad k<150\\ 1+0.25\,1\sin(2\pi k/25) & \text{for}\quad 150\le k<250\\ 1+0.51\sin(2\pi k/25) & \text{for}\quad k\ge 250\end{cases}. \tag{30}$$

The goal set G is defined by (14), where $\eta_0 = 0.02$ and the testing cost function has the following form:

$$J_T\left(\mathbf{y}_{NA,\tilde{\mathbf{v}}}(k),\mathbf{y}(k)|k\in K\right)=\frac{\sum_{k=1}^{500}\left(\mathbf{y}_{NA,\tilde{\mathbf{v}}}(k)-\mathbf{y}(k)\right)^2}{\sum_{k=1}^{500}\left(\mathbf{y}(k)\right)^2}\times 100\%, \tag{31}$$

where $\tilde{\mathbf{v}}$ is a vector of network parameters resulted in the training process.

Table I shows the exploration of the architecture space of the A*-algorithm. $s^{\left(o_i|i=1,\ldots,s\right)}$ denotes that there are s units with the IIR filters of $\left(o_i|i=1,\ldots,s\right)$ orders in the hidden/output layer. The architecture 1: $5^{(2,2,2,1,1)}$:$1^{(2)}$ is the nearest goal NA from the initial NA_0.

Table I: Optimal exploration of the architecture space by the A*-algorithm.

	1	2	3	4	5	6	7	8
hidden.	-	-	-	$1^{(0)}$	$1^{(1)}$	$1^{(2)}$	$2^{(2,0)}$	$2^{(2,1)}$
output	$1^{(0)}$	$1^{(1)}$	$1^{(2)}$	$1^{(2)}$	$1^{(2)}$	$1^{(2)}$	$1^{(2)}$	$1^{(2)}$

	9	10	11	12	13	14	15	16
hidden	$2^{(2,2)}$	$3^{(2,2,0)}$	$3^{(2,2,1)}$	$4^{(2,2,1,0)}$	$4^{(2,2,1,1)}$	$4^{(2,2,2,1)}$	$5^{(2,2,2,10)}$	$5^{(2,2,2,1,1)}$
output	$1^{(2)}$	$1^{(2)}$	$1^{(2)}$	$1^{(2)}$	$1^{(2)}$	$1^{(2)}$	$1^{(2)}$	$1^{(2)}$

The experiments show that the proposed method allows one to obtain an optimal or suboptimal architecture of the NDN. However, the efficiency of the architecture choice is dependent on a training process quality. Unfortunately, the training process of the NDN which is used as a model of dynamic system, seems to be an optimisation problem which is intrinsically related to a very rich topology of the sum square-error function. The EDBP algorithm usually finds one of the unsatisfactory local optima (the obtained $\tilde{\mathbf{v}}$ significantly differs from the optimal $\mathbf{v}^*$) and the calculated heuristic function $h(NA)$ introduces false information. Thus, after some re-runs, different optimal paths in the search space and different resulted network architectures have been obtained. The realisation described above is the best one. Thus algorithms of global optimisation, e.g. evolutionary algorithms or simulated annealing, should be applied as a training algorithms. One of the main disadvantages of these methods is their time-consuming processing.

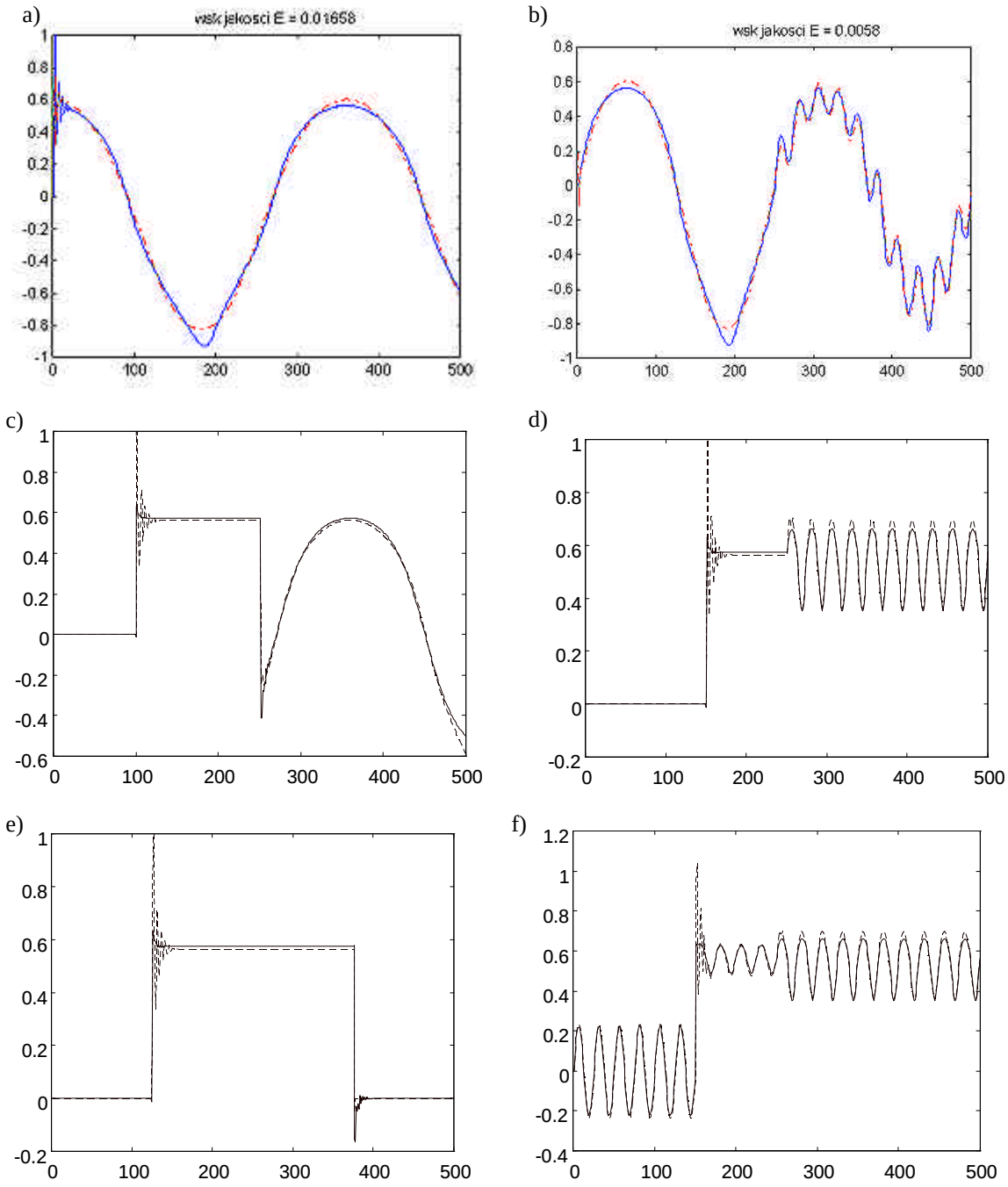

Figure 4: Responses of modelled system (solid line) and the resulting network (dotted line) on testing signals: a) (25) J_T = 1.66%, b) (26) J_T = 0.58%, c) (27) J_T = 0.71%, d) (28), J_T = 0.53% (29) J_T = 0.63%, f) (30) J_T = 0.55%.

CONCLUSION

In this work an heuristic search technique approach to find a minimal NDN architecture and thus to maximise the generalising ability of the trained network is proposed. This approach is based on the standard A*-algorithm and does not imply any restrictions with respect to the generated architectures. Optimal use of the heuristic information can be proved for rather strong restrictions concerning the network training, e.g. the training procedure should guarantee that

the global optimum in the space of network parameters is achieved. In practice, these conditions cannot be satisfied, especially when a training algorithm (like EDBP) bases on gradient descent methods.

On the other hand, the A*-algorithm is associated with computational efford that grows exponentially with the size of the goal architecture. This disadvantage is especially significant in the case of the NDN, where each DNM unit with the IIR of the n-th order is described by $2n+2$ parameters. Thus, applicability of the A*-algorithm in the case of complex non-linear dynamic systems is strongly limited.

Above problems determine our plans for further research.

REFERENCES

Ayoubi M., 1994, „Fault diagnosis with dynamic neural structure and application to a turbocharger", IFAC Symposium on Fault Detection, Supervision and Safety for Technical Process SAFEPROCESS'94, Espoo, Finland, pp. 618-623.

Cybenko G., 1989, „Approximation by superposition of a sigmoidal function", Math. Contr. Signals and Syst., Vol. 2, pp. 303-314.

Doering A., Galicki M., Witte H., 1997, „Structure optimisation of neural networks with A*-algorithm", IEEE Transactions on Neural Networks, Vol.8, No. 6, pp. 1434-1445.

Draye J.-P.S., Pavisic D.A., Cheron G.A., Libert G.A., 1996, „Dynamic recurrent neural networks: A dynamical analysis", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol. 26, pp. 692-706.

Elman J.L., 1990, „Finding structure in time", Cognitive Science, Vol. 14, pp. 179-211.

Fahlman S.E., Lebiere C.D., 1990, „The cascade-correlation learning architecture", In: „Advances in NIPS2", Ed. D. Toutetzky, pp.524-532.

Hecht-Nielsen R., 1990, „Neurocomputing", Reading: Addison-Wesley Publishing Company.

Hertz J., Krogh A., Palmer R.G., 1991, „Introduction to the Theory of Neural Computation", Redwood: Addison-Wesley Publ. Comp.

Hornik K., Stinchcombe M., White H., 1989, „Multilayer feedforward networks are universal approximators", Neural Networks, Vol. 2, No 5, pp.359-366.

Korbicz J., 1995, „Neural network architectures used in modelling/identification and control of dynamic systems", 8th International Conference on System Modelling Control, Zakopane, Poland, Vol. 3, pp. 54-59.

Korbicz J., Obuchowicz A. Patan K., 1998, „Network of dynamic neurons in fault detection systems", IEEE International Conference on System, Man and Cybernetics, San Diego, USA, pp.1862-1867.

Leontaritis I.J., Bilings S.A., 1985, „Input-output parametric models for non-linear systems", Int. J. of Control, Vol. 41, pp.303-344.

Narendra K.S., Parthasarathy K., 1990, „Identification and control of dynamical system using neural networks", IEEE Transactions on Neural Networks, Vol.1, pp. 12-18.

Nilson N., (1980), „Principals of Artificial Intelligence", New York: Springer-Verlag.

Obuchowicz A., 1999, „Training of the network of dynamic neurons using evolutionary search with soft selection", 8th International Symposium on Intelligent Information Systems, Ustroñ/Wis³a, Poland, (submitted).

Obuchowicz A., Korbicz J., 1998, „Evolutionary search with soft selection and forced direction of mutation", 7th International Symposium on Intelligent Information Systems, Malbork, Poland, pp.300-309.

Obuchowicz A., Patan K., 1998, „Network of dynamic neurons as a residual generator. The architecture optimization", 3rd National Conference on Diagnosis of Industrial Processes, Jurata, Poland, pp. 101-106 (in Polish).

Obuchowicz A., Politowicz K., 1997, „Evolutionary algorithms in optimization of a multilayer feedforward neural network architecture", 4th International Symposium on Methods and Models in Automation and Robotics, Miêdzyzdroje, Poland, Vol. 2, pp. 739-743.

Tsoi A.Ch., Back A.D., 1994, „Locally recurrent globally feedforward networks: A critical review of architectures", IEEE Transactions on Neural Networks, Vol. 5, pp. 229-239.