Variable encoding of modular neural networks for time series prediction

Bernhard Sendhoff, Martin Kreutz

1999

Variable encoding of modular neural networks for time series prediction

Bernhard Sendhoff
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany
bernhard.sendhoff@neuroinformatik.ruhr-uni-bochum.de

Martin Kreutz
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany
martin.kreutz@neuroinformatik.ruhr-uni-bochum.de

Abstract- The combination of evolutionary algorithms and neural networks for the purpose of structure optimization has frequently been discussed. In this paper we apply an indirect encoding method, the recursive encoding combined with a gradual growth process of the network structure, to the problem of time series prediction and modelling. Modularity of the network structure, the optimization of the encoding parameters on a larger time-scale, i.e. a meta-evolutionary process and the choice of encoding dependent search operators to enhance the strong causality of the search process will be discussed.

1 Introduction

Demanding modularity for the design of distributed information processing systems, especially neural networks, is desirably for a number of reasons. Finding an appropriate structure for a complex problem without imposing any constraints is likely to be a formidable task due to the high-dimensional and multi-modal search space. Furthermore, using a modular approach it is possible to obtain a growing neural network both with respect to the network complexity and the difficulty of the task. That is, the network can solve simplified problems even at an early developmental stage [1]. More generally speaking, a modular network design is a first step towards an automated problem decomposition and therefore to an hierarchical solution of a given task. In the evolutionary optimization of neural networks a modular approach is strongly connected to an indirect encoding of the network structure. Several proposals for the design of such an encoding have been put forward in the literature, e.g. [2, 3, 4]. In [3] a modular structure is imposed on the network from a-priori knowledge of the problem decomposition. This is questionable since it relies on the availability of sufficient knowledge about the problem to be able to design the optimization process accordingly. In [5] the modular network structure relies on the concept of automated function definition in genetic programming. The recursive encoding used in our work (Section 2) is an extension of the grammar encoding by Kitano [2], which inherently favours a modular network structure, but which at the same time can equally code for a complete random structure. In addition, the recursive encoding method provides the initialization of the network weights. In each generation each network is trained for a number of cycles using standard backpropagation. The encoding itself is variable and optimized on a larger time-scale. We introduce an extension of the recursive encoding which accomplishes a gradual growth process of the network structure during ontogeny. One of the aims of this work is to observe the development of modularity of the network structure and of the encoding parameters during optimization for two qualitatively different problems from the domain of time series prediction. In evolutionary algorithms the search operators must be chosen according to the encoding in order to meet the different constraints for a successful optimization process. In Section 3, we define specific mutation and crossover operators that enhance the strong causality (see e.g. [6, 7]) of the search process, which represents one of these constraints. In addition, the performance of the optimized networks is compared to randomly chosen neural structures which are averaged over several weight initializations.

2 Outline of the encoding

2.1 The recursive encoding method

In the recursive encoding, the connection matrix of a feed-forward network without a layered structure is developed in different stages. The elements of the connection matrix define the initial weights between neurons; in the case of a zero element the neurons are not connected. Since the matrix is feed-forward, only the upper triangular part is needed for the specification of the network. The diagonal elements define the threshold values for each neuron. The developmental process of the matrix is specified by a mapping from a first vector S_C (small chromosome) with integer elements to a second vector L_C (large chromosome), also with integer elements. In each step, each element of the connection matrix is replaced by a 2×2 matrix of new elements, starting with the first element of S_C which is replaced by the first four elements of L_C. In Figure 1 the growth of the connection matrix is shown. The following notation will be used for the recursive encoding:

$a^k_{\mu\nu}$	element at position (μ, ν) of the connection matrix in the kth recursion step. If the element is non-zero, it represents the initial weight between neuron μ and ν ($a^k_{\mu\nu} \in \{0, \ldots, N_{sym}\}$)
R	the number of recursion steps
k	the recursion step, ($k = 1, \ldots, R$)
N_{sym}	maximum integer number permitted in (S_C, L_C)

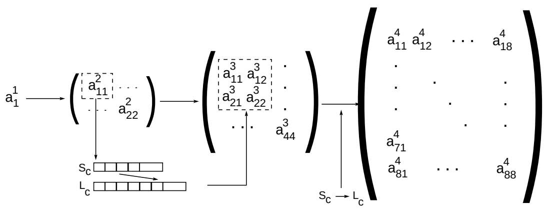

Figure 1: Scheme of the recursive development of the connection matrix up to a size of 8×8.

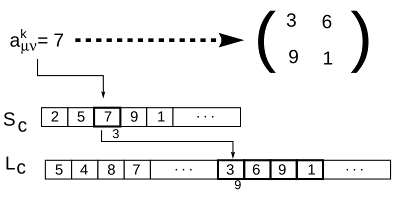

Figure 2: One element is replaced by four elements in one step in the recursive encoding.

$N(a^k_{\mu\nu})$	the index of the first element in S_C, which is identical to $a^k_{\mu\nu}$
d_{S_C}, d_{L_C}	dimension of the vector S_C, L_C (small,large chromosome)
s_i, l_i	elements of the vector S_C and L_C

At each recursion step k, the index $N(a^k_{\mu\nu})$ of the *first* element in S_C, which is identical to the connection matrix element $a^k_{\mu\nu}$, is determined; for example, index $N(a^k_{\mu\nu} = 7) = 3$ in Figure 2. The element is then replaced by the four elements at the positions

$$\big(\, 4 \cdot (N(a^k_{\mu\nu}) - 1) + 1, \quad 4 \cdot (N(a^k_{\mu\nu}) - 1) + 2, \quad 4 \cdot (N(a^k_{\mu\nu}) - 1) + 3, \quad 4 \cdot (N(a^k_{\mu\nu}) - 1) + 4 \,\big) \tag{1}$$

in the large chromosome L_C. Figure 2 shows the replacement of an element $a^k_{\mu\nu} = 7$ by the four elements $(3, 6, 9, 1)$ at the positions $(9, 10, 11, 12)$. Should $a^k_{\mu\nu}$ not be identical to any element of S_C, it is replaced by four so-called terminal symbols (in the notation of integer strings, the most convenient choice is zero). The terminal symbol (zero) denotes that no connection exists between neuron μ and ν. The terminal symbol is in turn always replaced by another four terminal symbols in a recursion step. Finally, the connection matrix is simplified by deleting all neurons which do not contribute to the network output. Thus, the main components of the recursive encoding are:

- ▷ In each step, each element of the connection matrix is replaced by a 2×2 matrix whose elements are specified by equation (1).
- ▷ If the element is not identical to any entry of S_C, the elements of the 2×2 matrix are given by $(0, 0, 0, 0)$.
- ▷ In each step, 0 is replaced by $(0, 0, 0, 0)$.
- ▷ R steps are carried out.

The recursive encoding has several advantages:

- ▷ The development of the connection matrix with integer numbers allows the initialization of the weights without the need of further extensions.
- ▷ The interpretation of the encoding as a mapping between two vectors makes theoretical analysis possible [8].
- ▷ The encoding scheme C is governed by the set of three parameters

  $$C = (R, d_{S_C}, N_{sym}), \tag{2}$$

  the number of recursion steps R, the dimension of the small chromosome d_{S_C} and the maximum integer number permitted N_{sym}. Thus, the encoding itself can be subjected to evolution on a larger time-scale. The evolution of the encoding (the genotype-phenotype map in biological terms) in evolutionary algorithms has recently attracted attention.
- ▷ At every step, the connection matrix can be used to define a neural network. In this way, it is possible to couple dynamically the developmental process, which is determined by the genetic information, with the learning process, see [1].

In its simplest form the recursive encoding develops a connection matrix from a vector S_C with elements $s_i \in \{1, \ldots, N_{sym}\}$ and a vector L_C with elements $l_i \in \{1, \ldots, N_{sym}\}$ with the described mapping. The connection matrix is then translated into a neural network in the following way. If $a^R_{\mu\nu} = 0$, neuron μ and neuron ν are not connected. All other integer values $a^R_{\mu\nu} \in \{1, \ldots, N_{sym}\}$ are mapped to an interval $[-\delta, \delta]$. These values constitute the initial weight values for the connection between neuron μ and neuron ν. In the final step, neurons, which are not connected to any other neurons and which do not serve as output neurons, are removed. If the system, which is modelled by the neural network, is of the form $\vec{y} = \vec{f}(\vec{x})$ with $\vec{x} \in I\!R^n$ and $\vec{y} \in I\!R^m$, then the first n neurons in the connection matrix are input neurons and the last m neurons are output neurons. The threshold function of each neuron is $tanh(x)$. Thus, the output of neuron ν (initially) is given by (x_ν and w_ν denote the input and input weights, respectively. ${a'}^R_{\mu\nu}$ are the matrix elements mapped to the interval.):

$$y_\nu = \tanh\left(\sum_{\mu=1}^{\nu-1} {a'}^R_{\mu\nu} y_\mu + {a'}^R_{\nu\nu}\right), \; \nu > n \tag{3}$$

$$y_\nu = \tanh\left(\sum_{\mu=1}^{\nu-1} {a'}^R_{\mu\nu} y_\mu + w_\nu x_\nu + {a'}^R_{\nu\nu}\right), \; \nu \le n. \tag{4}$$

This *basic* scheme has been successfully applied to network optimization for time series prediction by Sendhoff and

Kreutz [9]. However, it is not complete, i.e. not all possible network structures can be coded for. In order to guarantee completeness, it is sufficient to permit the terminal symbol 0 in L_C, $l_i \in \{0, \dots, N_{sym}\}$. The drawback, however, is that the encoding length can in the worst case exceed the encoding length of the direct encoding. The alternative is to allow elements in S_C and L_C which are not terminals but also code for a *no connection* entry between neurons in the connection matrix. This can be achieved by extending the set of possible elements in S_C and L_C to

$$s_i, l_i \in \{-N_{sym}^{neg}, \dots, -1, 1, \dots, N_{sym}\}, \qquad (5)$$

where all negative numbers are set to zero (no connection) in the final matrix. During the developmental steps, the negative integers are replaced in the same way as the positive integers. Further details of the completeness analysis of the recursive encoding con be found in [8]. Both variants remove some of the appealing simplicity of the recursive encoding. Although the question whether completeness is important for structure optimization cannot be answered, some remarks can however be made. If the purpose is to analyze modularity in the framework of information processing in systems like neural networks, completeness is unlikely to play a major role. On the other hand, it is likely that for any problem, which is to be solved by artificial neural networks, several structures are available which show similar performance, especially for extended systems. In the experiments carried out by the authors, the extensions of the recursive encoding method showed the best performance on all time series prediction problems and the results shown in Section 4 have been achieved with the extended scheme.

2.2 Gradual network growth in the recursive encoding

In order to highlight the growth process of the neural structure and to make the matrix replacement more gradual, thereby enhancing the strong causality [7] of the developmental phase, a variation of the original replacement scheme is introduced. The matrix at step $(k-1)$ is divided into four equally sized quadratic matrices. Only the elements from the matrices, which form the lower triangular part of the connection matrix at step $(k-1)$, are used for the mapping. These define, via the recursive mapping, the elements of the three matrices which form the lower triangular part of the connection matrix at step k. Each of these has the same dimension as the whole matrix at step $(k-1)$. The upper left matrix is built by copying the whole connection matrix of step $(k-1)$. The replacement scheme is visualized in Figure 3. An alternative albeit very similar method is to copy the network matrix from one developmental step before onto the upper left part of the connection matrix at the current step. Both methods coincide when no neurons of the network are removed. If neurons are removed, the copying process guarantees that the whole network structure of the previous step is included. In the experiments this growth process will be used, however, both methods hardly show any difference both with respect to the resulting network structure and to the performance. In order to investigate the combination of learning and genetically determined development, it is necessary to further extend the replacement scheme, by including weights which have been learned in previous steps, details can be found in [8].

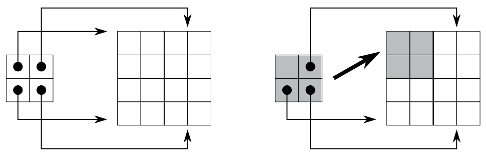
Figure 3: Left: The replacement scheme of the basic recursive encoding. Right: The extension to a more gradual growth process of the matrix. The matrix at step $(k-1)$ is transferred to the first quadrant of the matrix at step k, only the elements of the lower triangular matrix at step $(k-1)$ are used in the mapping.

3 Controlled genotype variations in structure optimization

The concept of strong causality [6, 7] relates distances on the genotype space to distances on the phenotype space in evolutionary algorithms. The claim is that the neighbourhood structure in genotype space should be conserved under the mapping from genotype to phenotype space, thus it represents a condition which the encoding and the mutation operator should fulfil. The causality condition can therefore be expressed as follows: *Small variations on the genotype space due to mutation imply small variations in the phenotype space.* In [7] it was quantified using a definition of distances on the genotype space, which is dependent on the mutation operator. The standard versions of the genetic algorithm and the evolution strategy for parameter optimization were analytically examined with respect to the causality condition in [8]. The recursive encoding method in general does not fulfil this condition, it is therefore necessary to examine whether changes to the operators can enhance the degree to which strong causality is fulfilled. In the following, we will empirically analyze (with respect to the performance and the dynamics of the search process) specific mutation and crossover operators, which have been shown to increase quantitatively the degree of causality [7].

Operator set A: The mutation probability for the chromosomes S_C and L_C is position independent, the values are given in Table 1. Uniform crossover has been applied to both chromosomes, that is, each element has an equal and independent probability of being inherited from either of the parents. If the vectors have different dimensions, the remaining positions are copied from the longer vector. If the offspring vector dimension is larger than that of either of its parents, the remaining positions are drawn uniformly from the set of symbols.

parameter	**value**	**parameter**	**value**
learning rate	0.01	momentum factor	0.5
max. d_{S_C}	50	max. N_{sym}	50
max. R	5	min. net size	6
weight interval	$[0.8, -0.8]$	max net size	64
$p_m(S_C)$	$2 \cdot (d_{S_C})^{-1}$	$p_c(S_C)$	0.75
$p_m(L_C)$	$2 \cdot (d_{L_C})^{-1}$	$p_c(L_C)$	0.75
$p_m(C)$	0.05	$p_c(C)$	0.1

Table 1: Parameter values for the experiments A & B to analyze the influence of the changes in the mutation and the crossover operator.

Operator set B: The mutation probability is position and symbol dependent, based upon the values specified in Table 1. Since the first four symbols in S_C and in L_C have a strong influence on the encoding, their mutation probability is reduced to $p_m(B) = p_m^2(A)$. Furthermore, the mutation probability of symbols which occur more than once is increased, since they do not influence the encoding. If a symbol occurs n times in the chromosome S_C, the mutation probability is given by $p_m(B) = n \cdot p_m(A)$, with a cut-off at $p_m(B) = 0.9$. The crossover on L_C is uniform with respect to four-tuples of elements, that is, for each set of four elements it is determined only once from which parent the symbols are copied. This reflects the functional unity of four-tuples representing 2×2 matrices in the recursive encoding.

In order to analyze the influences of these changes two experiments A & B have been carried out predicting the Lorenz time series, see Section 4, one step ahead. Deterministic selection was employed with $(\mu, \lambda) = (20, 60)$ over 150 epochs. The parameters for the encoding, the network and the learning algorithm, which is a standard back-propagation algorithm are shown in Table 1. p_m and p_c denote the mutation and crossover probability of the different chromosomes, note that C contains all values of the encoding; see equation (2). The initial weight values were fixed by the evolutionary algorithm and mapped to the interval $[-0.8, 0.8]$. Experiment A uses the standard operator set A and experiment B the operator set B, as described above. For each epoch the differences in the fitness values, the genotype and the phenotype structures were recorded using the following measures. Since genotypes and phenotypes have variable length, equations (9,10) look slightly confusing, however they only measure the average difference. All quantities were averaged over the whole population in each epoch. Let $s_i^{p_1,p_2}$, s_i^o, $l_i^{p_1,p_2}$ and l_i^o denote the genotype vector components for the parents and the offspring, $a_{\mu\nu}^{p_1,p_2}$ and $a_{\mu\nu}^o$ the matrix entries of the phenotype and f^{p_1,p_2} and f^o the fitness values. Let

$$d_{l,s,a}^{min(k)} = \min(d_{l,s,a}^{p_k}, d_{l,s,a}^o) \qquad (6)$$

$$d_{l,s,a}^{max(k)} = \max(d_{l,s,a}^{p_k}, d_{l,s,a}^o) \qquad (7)$$

be the minimal and maximal vector dimensions and the minimal and maximal matrix dimension, respectively. The set $\mathcal{M}^{no(k)}$ contains all indices of the matrix entries which do not overlap in the parent and offspring matrices. The matrix entries are denoted by $a_{\mu\nu}^{no(k)}$ and refer to the parent or offspring matrix entries depending on which one has the larger dimension. The notation for the genotype vectors is analogue $s_i^{no(k)}$, $l_i^{no(k)}$.

$$\mathcal{D}_{fit} = \frac{1}{2}\left(\frac{|f^{p_1} - f^o|}{f^{p_1}} + \frac{|f^{p_2} - f^o|}{f^{p_2}}\right) \qquad (8)$$

$$\begin{aligned}\mathcal{D}_{pheno} = \frac{1}{2}\Bigg(&\sum_{\mu\nu}^{d_a^{min(1)}} |a_{\mu\nu}^{p_1} - a_{\mu\nu}^o| + \sum_{\mu\nu \in \mathcal{M}_{no(1)}} |a_{\mu\nu}^{no(1)}| \\ &+ \sum_{\mu\nu}^{d_a^{min(2)}} |a_{\mu\nu}^{p_2} - a_{\mu\nu}^o| + \sum_{\mu\nu \in \mathcal{M}_{no(2)}} |a_{\mu\nu}^{no(2)}| \Bigg) \qquad (9)\end{aligned}$$

$$\begin{aligned}\mathcal{D}_{geno} = \frac{1}{2}\Bigg(&\sum_{i=0}^{d_s^{min(1)}} |s_i^{p_1} - s_i^o| + \sum_{i=d_s^{min(1)}}^{d_s^{max(1)}} |s_i^{no(1)}| \\ &+ \sum_{i=0}^{d_l^{min(1)}} |l_i^{p_1} - l_i^o| + \sum_{i=d_l^{min(1)}+1}^{d_l^{max(1)}} |l_i^{no(1)}| + \sum_{i=0}^{d_s^{min(2)}} |s_i^{p_2} - s_i^o| \\ &+ \sum_{i=d_s^{min(2)}+1}^{d_s^{max(2)}} |s_i^{no(2)}| + \sum_{i=0}^{d_l^{min(2)}} |l_i^{p_2} - l_i^o| + \sum_{i=d_l^{min(2)}+1}^{d_l^{max(2)}} |l_i^{no(2)}| \Bigg)\end{aligned} \qquad (10)$$

The results are shown in Figure 4 (a)–(e). Firstly, it can be seen that the operator set B clearly shows better results with respect to the performance of the algorithm both for the best individual (a) and the average fitness (b). The fitness is given by the *rms* error averaged over the test data set. The task is, therefore, to minimize the fitness. The overall conclusion from the recorded genotype, phenotype and fitness differences is that the operator set B allows smaller steps in all spaces. This is most profound in the genotype space for which changes are directly influenced by the mutation operator. $\mathcal{D}_{geno}$ assumes distinctly smaller values, especially between epochs 40–80. This also corresponds to a drop in both $\mathcal{D}_{pheno}$ and $\mathcal{D}_{fit}$, although less clearly than in the case of the genotype differences. However, the differences between experiments A and B for $\mathcal{D}_{fit}$ are greatest between epochs 40–80. This corresponds to the course of the fitness, for which both the best and the average values in exp. A stagnate from epoch 40 onwards. Whereas in simulation B, the fitness evolves towards better values between epochs 40–80 and beyond epoch 80 or 90 it remains unchanged. Operator set B was used for the experiments performed in the next section.

4 Time series prediction and modelling

We applied the evolutionary algorithm (based on experiment B last section) with the proposed encoding to the problem

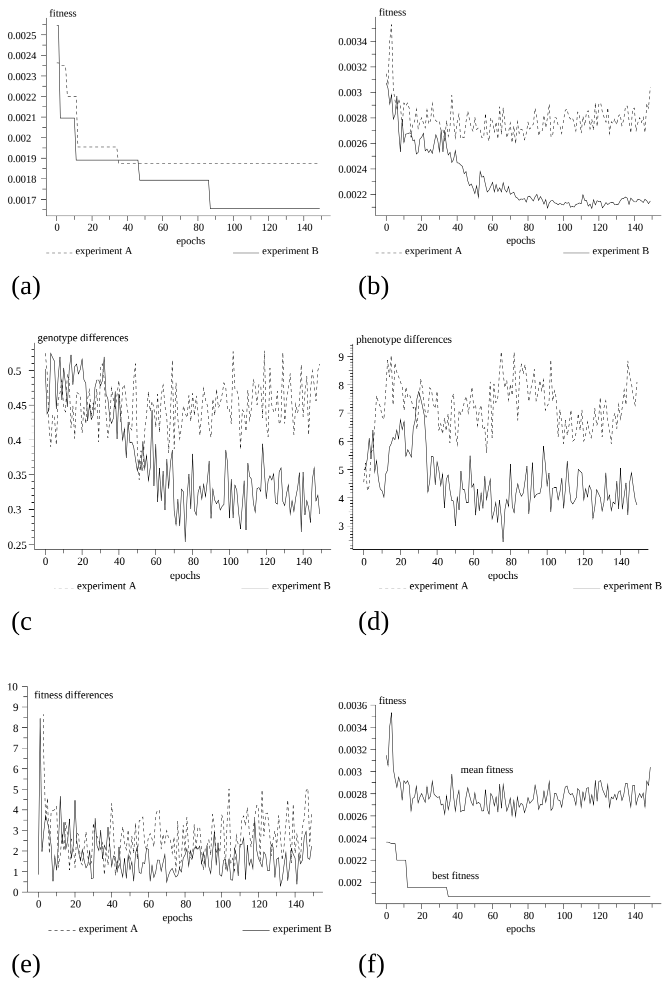

Figure 4: The best fitness values (a) and the average fitness values (b) for both simulations. Figures (c)–(e): the genotype differences, $\mathcal{D}_{geno}$, equation (10), the phenotype differences, $\mathcal{D}_{pheno}$, equation (9) and the fitness differences, $\mathcal{D}_{fit}$, equation (8) for each epoch averaged over the whole population. The best and the average fitness (f) for a third simulation with higher mutation rate.

of one step prediction (experiment B_1) and five step iteration (experiment B_2) of the chaotic Lorenz system [10]:

$$\frac{dx(t)}{dt} = -y(t)^2 - z(t)^2 - a\,x(t) + a\,F \qquad (11)$$

$$\frac{dy(t)}{dt} = x(t)\,y(t) - b\,x(t)\,z(t) - y(t) + G \qquad (12)$$

$$\frac{dz(t)}{dt} = b\,x(t)\,y(t) + x(t)\,z(t) - z(t). \qquad (13)$$

The differential equation system (11–13) was solved numerically with the Runge-Kutta 4th order method with an integration step $\Delta t = 0.05$ for 10000 time steps for the parameter setting ($a = 0.25, b = 4.0, F = 8.0, G = 1.0$). The attractor dimension for this setting is $d_A \approx 2.5$. The one-step prediction task is a straightforward function approximation problem for which neural networks have been frequently applied successfully.

The five step iteration task requires that neural networks capture some of the dynamics of the system. The model receives the system state $\vec{x}(t)$ at time t as input and iterates for five or more general for m time steps. During the iteration, the network uses the prediction $\vec{\tilde{x}}(t_0 + n\,\Delta t)$ of the systems state as input for the prediction at step $n+1$, $\vec{\tilde{x}}(t + (n+1)\,\Delta t)$. The iteration task is very difficult for the neural network, since there are two sources of error. Firstly, during the iteration the model itself constitutes a dynamical system. If the dynamic of the model is very different (this can be the case even if the one step prediction shows very good results) the predicted orbit will rapidly depart from the true orbit. For example, if the network has a fix-point dynamic it might only need a few iterations to reach this point. This effect can be seen in poorly trained networks for larger iteration times. Secondly, due to the irregularity of the system, the errors of the model in each time step will be amplified even if it perfectly captures the dynamic of the original system. Learning the underlying dynamics is difficult for purely feed-forward neural networks [11]. The learning algorithm usually has to be modified to include the iteration task in order to achieve good results [12, 13]. Note that the first source of error is the problem of modelling a system and cannot be avoided, whereas the second problem can be circumvented, if in addition to the standard error measure invariants of the dynamics are used to determine the performance of the network in modelling the original system [13, 14]. However, due to the additional time for the evaluation of the Liapunov exponents this approach is not feasible for structure optimization.

In both experiments, the error measure was the standard root mean squared (*rms*) error on the validation data set. The training set was used for the network training (once after the developmental process) and the test set as an early-stopping criterion, which however was hardly used because of the relatively few training cycles and the small sample size (all three sets consist of 500 data). In the evolutionary algorithm the operators, which were discussed in the last section, were applied together with deterministic selection, which turned out to deliver empirically the best results. All parameters of the evolutionary algorithm and of back-propagation learning are summarized in Table 2. Each individual consists of four chromosomes (S_C, L_C, C, *input-weights*); the larger time-scale of C is realized by reducing the values of p_m and p_C.

The results are shown in Figure 5 and 7. In exp. B_1 the size of the small chromosome is consistently larger than the number of permitted symbols N_{sym} which favours larger network's with higher connection probability. The variations of the most influential structure parameter R is moderate even in the average case (not shown). Thus, the interruptive nature of this parameter is kept under control. In many instances changes in the fitness values are connected to changes of the structure parameters contained in C and/or the connectivity and the networks dimension. The fitness values of

parameter	**value**	**parameter**	**value**
learning rate	0.0175 (0.0009)	momentum factor	0.75
max. d_{S_C}	50	max. N_{sym}	50
max. R	7	min. net size	6
weight interval	$[1.0, -1.0]$	max net size	256
max. training time	120 sec.		
μ	20	λ	80
$p_m(S_C)$	$2 \cdot (d_{S_C})^{-1}$	$p_c(S_C)$	0.75
$p_m(L_C)$	$2 \cdot (d_{L_C})^{-1}$	$p_c(L_C)$	0.75
$p_m(C)$	0.05	$p_c(C)$	0.05
$p_m(input)$	0.33	$p_c(input)$	0.75

Table 2: Parameter values for the experiment (B_1) for the one step prediction and (B_2) for the five step iteration (values are given in brackets if different) of the Lorenz time series with the recursive encoding method and back-propagation learning. p_m and p_c denote the probabilities of mutation and crossover for the different chromosomes, S_C, L_C, the encoding C and the input weights $input$.

three different versions of the encoding are shown in Figure 5 (c) together with the average error values of various random network structures in Figure 5 (d). The dashed/dotted curve shows the prediction error for the basic recursive encoding, the dashed curve the encoding combined with a gradual development of the network structure and the interlaced curve the result of gradual development combined with negative integers in S_C and L_C to code for additional zero elements in the connection matrix. The introduction of the more gradual growth process during ontogeny increases the performance of the evolutionary algorithm; it seems that a more gradual development also allows a more gradual search process. Furthermore, allowing to encode for "no connection" entries with more than one symbol, also seems to be beneficial for the evolutionary algorithm, although the difference is not as pronounced as for the more gradual development. All different encoding methods compare favourably with the random structures depicted in Figure 5 (d). Quantitatively, compared to the best average performance of the random network structures, the best evolved structure shows a fitness (error) decrease by a factor of 0.18, Figure 5 (c) & (d). This corresponds to a performance increase by a factor of 5.5 (best evolved network $rms = 6.75 \cdot 10^{-5}$, best random network (av.) $rms = 3.71 \cdot 10^{-4}$). We observe further that the best random structures are smaller than the evolved structures, which is due to the random initialization process of the weights, which seems to play a decisive role in their performance (very large variance). In Figure 5 (b) the connection matrix[1] of the best network is shown. The different initial values of the weights are represented by different colours, a medium dark blue corresponds to a zero en-

[1]Only the upper triangular part of the connection matrix is needed for a feedforward network. However, the whole matrix at the various growth steps during the ontogenetic phase is used. Therefore, it is sensible not to restrict the analysis to the triangular matrix.

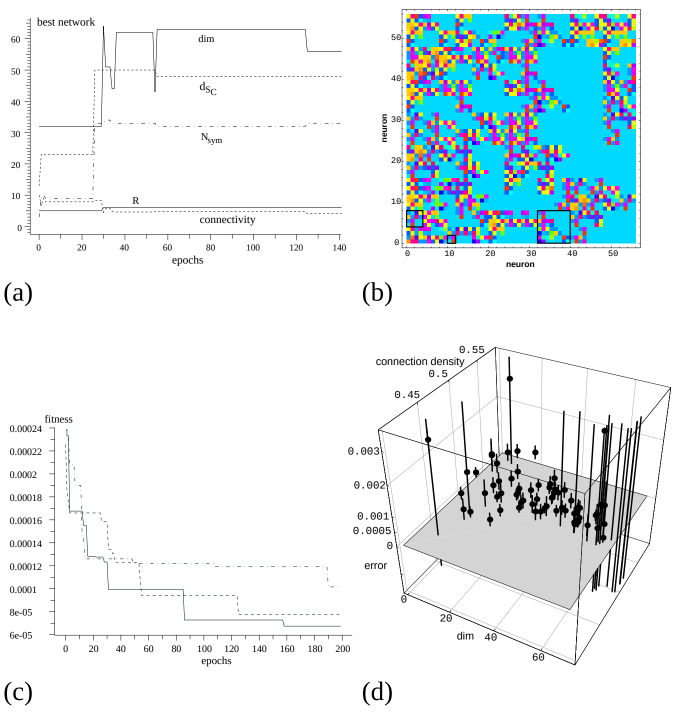

Figure 5: Results from experiment B_1. (a) Encoding and network parameters (the connectivity values have been multiplied by ten). (b) Connection matrix, the framed parts are the modules which appear most often. (c) Fitness values (rms on the val. set) for three different encoding versions; dashed/dotted curve – basic recursive encoding; the dashed curve – encoding combined with gradual development; interlaced curve – gradual development combined with negative integers in L_C, S_C. (d) Average error (50 trials) of random networks with variances (dim $\sim$ number of neurons).

try in the connection matrix[2] In the matrix identical patterns, which occur at different places, can be observed. In Figure 6 the sub-matrices, which appear with the highest frequency and their number of occurrences are shown. These patterns mark network structures and weight initializations, which are used more often than others. Although this modularity is inherent in the recursive encoding method it is not necessarily expressed. This is due to the adaptability of the encoding, which is evolved on a larger time-scale (on average the probabilities of changes is reduced by a factor of 5). It must therefore be beneficial for the optimization process to exploit this tendency of the encoding towards modularity. The number of experiments and the amount of data are not sufficient to conclude that these re-occurring patterns are suitable micro-structures for the chosen task. However, the conclusion that it is easier for the search process to build the network out of similar patterns can be reached. From Figure 7 for the iteration task, we first note that the resulting encoding and,

[2]The connection matrices are shown in colour and 3d in more detail on the web: http://www.neuroinformatik.ruhr-uni-bochum.de/ini/PEOPLE/bs/matrix.html.

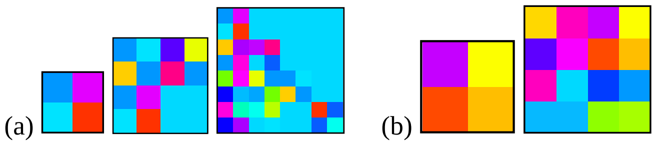

Figure 6: Pattern of the modules which re-occur with the highest frequency in the connection matrices for experiments B$_1$, Figure 5 (b) and B$_2$, Figure 7 (b). The 2×2 matrix in part (a) appears 53 times, the 4×4 and the 8×8 matrix, 12 and 3 times, respectively. In experiment B$_2$ the 2×2 and the 4×4 matrix in part (b) occur 4 and 2 times.

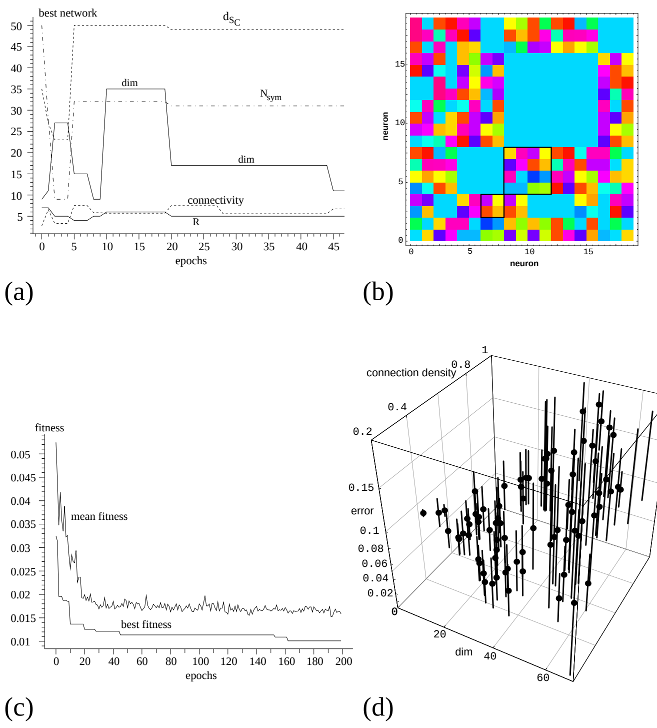

Figure 7: Results from experiment B$_2$. (a) Encoding and network parameters (the connectivity values have been multiplied by ten). (b) Connection matrix, the framed parts are the modules which appear most often. (c) Fitness values (rms on the val. set) of the best individual and the population mean. (d) Average error (50 trials) of random networks with variances (dim $\sim$ number of neurons).

especially the network structure, differ significantly from the one step prediction, although the parameters of the evolutionary algorithm have not been changed. The dimension of the network is smaller both for the best individual and on average. This is mirrored by the encoding parameter R which is decreased by one, corresponding to a decrease in neurons by a factor 2. The encoding parameters determine the overall direction of search. Smaller networks are either specified by small d_{S_C} and large N_{sym} values (during the first generations, Fig. 7 (a)) or by smaller R values (during later generations, Fig. 7 (a)). The best network structure is reduced

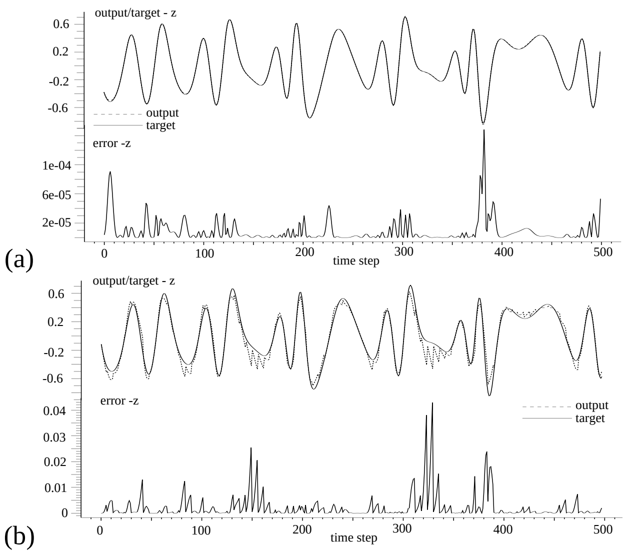

Figure 8: The z coordinate of the state vector (x, y, z) of the Lorenz 84 time series (target) and the one step prediction (a) and the five step iteration (b) of the best network (output) together with the squared error for each step.

from 32 neurons down to 20 neurons after the developmental procedure. At the same time, the modularity of the network is decreased. This decrease is not only due to the smaller network size, for experiment B$_1$, 1636 out of 3136 matrix entries belong to re-occurring pattern (52 %), whereas for B$_2$ the ratio is 132 to 361 (37 %). The reason for consistently choosing smaller networks lies in the difficulty of modelling the dynamic of the Lorenz system for large networks with limited number of data (training, test and validation sets consist of 500 pattern each). One might suspect that the initial weight setting for this process might be more important than for the one step prediction problem. Again for comparison the average error with variance of randomly chosen network structures for 50 random weight initializations are shown in Figure 7 (b). The variances of the network are further increased compared to Figure 5 (d) for the one step prediction problem, this is in accordance with the observations from the evolutionary optimization that the weight initialization seems to play a more important role for the five step iteration problem. The average error of the best network is larger than the error of the best evolved network, although the difference is not as pronounced as in the one step prediction problem. The best networks are the ones with a dimension between 20 and 40 and low connection densities.

The target time series, the network output and the error for each time step for one coordinate (z) are shown in Figure 8 for both the one step prediction (exp. B$_1$) and the five step iteration task (exp. B$_2$).

5 Summary and Conclusion

We presented the recursive encoding method combined with a gradual growth process of the network structure during ontogeny. The encoding method favours modular network structures and combined with specific search operators, which enhance the causality of the process, the optimized neural structures clearly outperform random structures on two qualitatively different problems from the domain of time series forecasting. Of course, the recursive encoding method can demonstrate its ability to find good[3] network structures best if larger networks are likely to be searched for. However, if we want to deal with larger networks efficiently, we need more appropriate learning rules than the different versions of gradient descent that is currently available. A local learning rule, which could be constrained to larger modules might be a possible direction to pursue. For such a rule the identification of modules is important not just with respect to their occurrence in the network structure but also to their role in the information processing structure of the whole system.

The growth process which was introduced in this paper also highlights the possibility of an ontogenetic stage in the optimization of neural structures. The implications are manifold. Firstly, it defines a system for the analysis of the dynamics between learning and evolution. Secondly, it might be one step towards the hierarchical solution of complex problems. The main idea is to allow the network structure to grow with the problem during ontogeny. Therefore, the problem would be solved step-by-step by a system whose capabilities would grow alongside. Of course, these questions are far from being solved, however, preliminary results have been obtained [8].

Acknowledgements

The authors would like to thank W. von Seelen, C. Igel and P. Stagge for stimulating discussions on the subject. The Höchstleistungsrechenzentrum Jülich, Germany is acknowledged for providing the computation time on the Cray T3E massively parallel supercomputer and for their expert advice. The project is supported by the German Ministry of research under grant SONN II 01IB701A0 and grant LEONET 01IB802C4.

[3]We have purposely used "good" instead of "optimal", since for large structures, due to the huge search space, finding a structure that works well will be all that we can hope for.

References

[1] B. Sendhoff and M. Kreutz. A model for the dynamic interaction between evolution and learning. *Neural Processing Letters*, 1999. Accepted.

[2] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476, 1990.

[3] F. Gruau. Genetic synthesis of boolean neural networks with a cell rewriting developmental process. In Darrell Whitley and J.D. Schaffer, editors, *Proceedings COGANN-92 International Workshop on Combination of Genetic Algorithms and Neural Networks*, pages 55–73. IEEE Computer Society Press, 1992.

[4] S.A. Harp, T. Samad, and A. Guha. Towards the genetic synthesis of neural networks. In J.D. Schaffer, editor, *Genetic Algorithms: Proceedings of the 3rd International Conferences (ICGA)*, pages 360–369. Morgan Kaufmann, 1989.

[5] F. Gruau. Automatic definition of modular neural networks. *Adaptive Behavior*, 3(2):151–183, 1995.

[6] I. Rechenberg. *Evolutionsstrategie '94*. Friedrich Frommann Holzboog Verlag, 1994.

[7] B. Sendhoff, M. Kreutz, and W. von Seelen. A condition for the genotype–phenotype mapping: Causality. In Thomas Bäck, editor, *Genetic Algorithms: Proceedings of the 7th International Conferences (ICGA)*, pages 73–80. Morgan Kaufmann, 1997.

[8] B. Sendhoff. *Evolution of Structures – Optimization of Artificial Neural Structures for Information Processing*. Shaker Verlag, Aachen, 1998. Doctoral dissertation.

[9] B. Sendhoff and M. Kreutz. Evolutionary optimization of the structure of neural networks using a recursive mapping as encoding. In G.D. Smith, N.C. Steele, and R.F. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms – Proceedings of the 1997 International Conference*, pages 370–374. Springer Verlag, 1998.

[10] E.N. Lorenz. Irregularity: A fundamental property of the atmosphere. *Tellus*, A(36):98, 1984.

[11] J. Principe, A. Rathie, and J. Kuo. Prediction of chaotic time series with neural networks and the issue of dynamic modeling. *International Journal of Bifurcation and Chaos*, 2(4):989, 1992.

[12] J.C. Principe and J.–M. Kuo. Dynamic modelling of chaotic time series with neural networks. In R.P. Lippmann, J.E. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 7. Morgan Kaufmann, 1995.

[13] G. Deco and B. Schürmann. Neural learning of chaotic dynamics. *Neural Processing Letters*, 2(2):23–26, 1995.

[14] P. Stagge and B. Sendhoff. An extended elman net for modeling time series. In W. Gerstner, A. Germond, M. Hasler, and J.- D. Nicoud, editors, *Artificial Neural Networks (ICANN'97)*, volume 1327 of *Lecture Notes in Computer Science*, pages 427–432. Springer Verlag, 1997.