Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Optimization for Problem Classes – Neural Networks that Learn to Learn

Michael Hüsken, Jens Gayko, Bernhard Sendhoff

2000

Preprint:

This is an accepted article published in IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks. The final authenticated version is available online at: https://doi.org/[DOI not available]

Proceedings of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (ECNN 2000),
IEEE Press, 2000

Optimization for Problem Classes – Neural Networks that Learn to Learn –

Michael Hüsken
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany
michael.huesken@uni-bochum.de

Jens E. Gayko
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany
jens.gayko@gmx.de

Bernhard Sendhoff
Future Technology Research (FTR)
HONDA R&D Europe GmbH
Carl-Legien-Strasse 30
63073 Offenbach/Main, Germany
ftrseb@ipm.net

Abstract- The main focus of the optimization of artificial neural networks has been the design of a problem dependent network structure in order to reduce the model complexity and to minimize the model error. Driven by a concrete application we identify in this paper another desirable property of neural networks which has to be the result of evolutionary optimization of the network structure – the ability of the network to efficiently solve related problems denoted as a class of problems. In a more theoretical framework the aim is to develop neural networks for adaptability – networks which learn (during evolution) to learn (during operation). The process of evolutionary optimization is time consuming, it is therefore also from the perspective of efficiency desirable to design structures which are applicable to many related problems. In this paper, two different approaches to solve this problem are studied, called ensemble method and generation method. We empirically show that an averaged Lamarckian inheritance seems to be the most efficient way to optimize networks for problem classes, both for artificial regression problems as well as for real-world system state diagnosis problems.

1 Introduction

Artificial neural networks (ANNs) are a widely used tool for nonlinear data analysis, especially multi-layer perceptrons (MLPs) are well suited for many tasks in function approximation, classification and nonlinear regression. The parameter values of ANN models are adjusted iteratively usually by some kind of gradient descent on an error surface, which is defined by an appropriate distance measure between the ANN's output and the target values. Since in principle the adaptation of the parameters (called learning following the biological paragon) is an optimization problem, it was natural to start to combine ANNs with evolutionary algorithms for weight optimization.

In the last decade nearly every aspect of ANNs has been subject to applying evolutionary algorithms to increase their modelling performance: parameter adaptation and initialization, specification of the structure or architecture, feature selection and rule extraction, see Yao (1999) for a recent survey of the literature. Most of the applications (and the approach in this work) of evolutionary algorithms to neural networks (EANN) have concentrated on the ANN's structure, i. e. the connectivity (which can include the weight initialization). Of course, also alternative methods of finding a problem dependent ANN structure have been put forward, most notably pruning (Reed 1993) and cascade correlation (Fahlmann and Lebiere 1990). At the same time ANNs have been combined with evolutionary algorithms at a more analytical level (as compared to pure performance increase) to achieve a better insight into network learning (note that in this area the paradigm "learning to learn" is used in a different context than in this paper) (Yao 1999) and into the interaction between evolution and learning (Nolfi, Miglino, and Parisi 1994; Sendhoff and Kreutz 1999).

However, despite the considerable interest in EANN, the application to real world problems has been sparse (with exceptions, see e. g. Gayko, Lohmann, Voss, Sendhoff, and Zamzow 1997) in particular compared to the large number of real world problems which have been tackled by ANNs with standard gradient learning. We believe that two reasons for this can be identified:

- The awareness of the influence of the structure of the ANN and the initialization of the weights on the ANN's performance might be small in particular among those people who are mainly interested in the application of ANNs. A concise comparison between the influences of the different characteristics of ANNs for relevant problems might be useful to enhance this awareness.
- The increase in the computation time to determine ANN characteristics by evolutionary methods is often regarded as too high compared to the expected benefits on one particular problem. However, if the evolutionary algorithm could be used to engineer an ANN for a number of problems in parallel this might change.

The work described in this paper addresses the second point. The robustness of ANN has been pointed out frequently in literature, in particular with regard to the generalization from one data set to another, which both stem from the same system. EANN might be able to take this "robustness" capability of ANNs one step further and to introduce a generalization between different problems , called *second order generalization*. Of course, these problems have to have some common aspects, which can be exploited for the process of

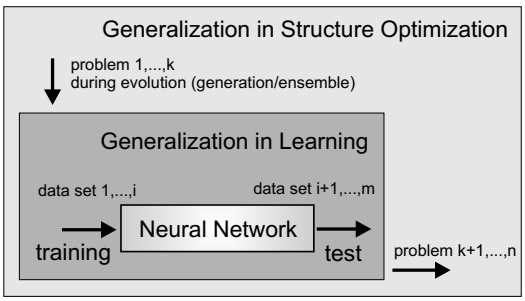

Figure 1: The proposed two different levels of generalization in ANNs: The first (inner) generalization is from one data set to another during network learning. The second (outer) generalization is from one problem to another (of one class) during evolutionary optimization.

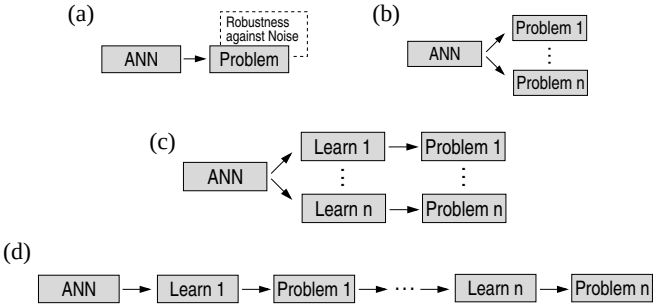

Figure 2: Different levels of robustness and different set-ups to simulate changing environments and/or problem classes

specifying an ANN's structure, which is appropriate for the whole *problem class*. These two steps of generalization are visualized in Figure 1.

Of course, one of the main questions is the definition of appropriate problem classes. This is an open problem of optimization in general and we will not be able to solve it in this work. However, problem classes can be defined from the applicational point of view;

[...] Problems which share some common features and should be solved at the same time are denoted as classes of problems. [...]

We will propose one artificial problem class in Section 3 which seems to be sensible for ANN modelling and present one class which is a result of a real world application and which triggered many of the ideas described in this paper.

The remaining paper is organized as follows. In the next section, we will embed the idea of problem classes into the more common notion of changing environments and outline some different approaches. In Section 3 we will describe both the artificial problem class and the real world application. The proposed methods and a comparison between different strategies will be presented in Sections 4 and 5. In the last section, we will summarize and discuss our findings.

2 Problem classes, changing environments and adaptability

The problem that an ANN should be able to adapt to changing environments during operation can be found in many applications. Indeed it can be argued that for real world problems stationarity is more an exception than a general rule. In principle multi-layer perceptrons are able to approximate nonlinear functions to arbitrary accuracy (Funahashi 1989), of course in practice the limiting factor is the needed computation time and the size of the data set. In many applications, the number of available data is small and particularly in safety sensitive applications (like in the automotive industry) the ANN must be able to adapt to changing constraints rapidly and reliably. In the next three paragraphs we will discuss different concepts of generalization between problems and outline possible EANN approaches to achieve such additional capabilities.

The lowest level of robustness of ANNs in changing environments is against noise, i. e. statistical fluctuations of some input parameters, see Figure 2 (a). At the same time, this can include fluctuations of ANN parameters, see e. g. Stagge (1998). The extension of theadaptability of the ANN to cope with more serious changes in the environment, i. e. with different problems which share some common characteristics, are shown in Figure 2. In Figure 2 (b) one network configuration has to be able to solve several different problems; this is unlikely to be realizable without considerable deterioration of the performance compared to specialized ANNs. This problem can be overcome by introducing an additional short period of learning, see Figure 2 (c). Starting from a common ANN structure each problem of one problem class can be learned in a short time compared with the learning time of a network with an arbitrary structure e. g. a fully connected one. In practice the most likely situation is depicted in Figure 2 (d): the ANN has to be able to *sequentially* adapt to changes in the environment, i. e. to different problems which all belong to the same class.

As depicted in Figure 2 (c) and (d) the crucial point for solving related problems from one class is adaptation. In this paper, we denote the ability of an ANN to adapt to a whole problem class as *second order adaptation*[1]. In this sense, during evolutionary optimization of the structure, the ANN *learns to learn* a certain class of problems. Of course, the benefit of using the EANN paradigm should be the reduction of computation time in this approach.

In principle there are two methods which can be used to realize ANNs, which are capable of fulfilling the demands described above. Each individual ANN can be tested in each generation on a number of problems from the same class to estimate the second order adaptability; this is depicted in Figures 3 (a) and we will call it the *ensemble method*. Another possibility is to change the task in different generations (the *generation method*, see Figure 3 (b)), i. e. to select the best individual in generation t for problem i_t and in generation

[1]Within this paper the additional term "second order" is often omitted. However, from the context it should be clear whether we refer to the "standard" generalization capability of ANNs or to the proposed second order generalization between problems.

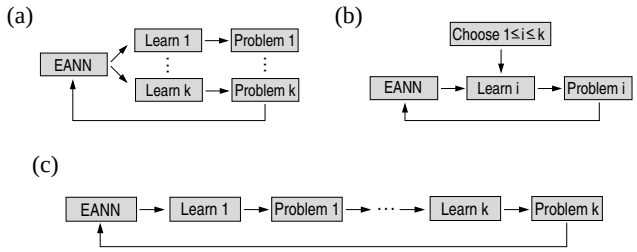

Figure 3: Different strategies to develop second order adaptive ANNs which can cope with changing environments.

$t+1$ for a different problem i_{t+1}, see e. g. Nolfi, Miglino, and Parisi (1994), Sasaki and Tokoro (1998). Both methods can claim to have a biological counterpart. Of course the natural environments are constantly changing and biological information processing systems (e. g. the brain and the immune system) must be able to cope with such situations (indeed the ability to learn at all is a means to cope with such changes on a short time-scale) during evolution. At the same time, the mammalian brain has to cope with a variety of changing and apriori completely unknown problems during life time.

In principle, the idea behind these thoughts is to optimize ANN with respect to their ability to process information without being confined to one specific problem. Thus, the quality function is adaptability. However, the nebulous meaning of the notion of "adaptability" (which is frequently used in literature) makes it difficult to precisely define a measure for it other than either with respect to concrete applications, like our "problem classes" approach, or with respect to very basic demands on information processing in distributed systems like the information processing at the edge to chaos paradigm (Crutchfield and Young 1989)[2].

After these more general remarks on the notions of changing environments, problem classes and adaptability, we will in the next section introduce an artificial problem class and one based on a real world application.

3 Real world and artificial problem classes for nonlinear regression

In this section, we introduce two different problem classes: a real world problem of machine condition monitoring and an artificial problem class. Whereas the first one is mainly utilized to show the need for adaptive ANNs, the latter is employed to investigate different methods for developing adaptive ANNs.

Both problems can be modelled using a feedforward ANN with a multilayer structure. In this investigation, we restrict ourselves to feedforward ANNs with one layer of hidden neurons, according to the universal approximation theorem (Funahashi 1989). The hidden neurons have sigmoidal activation functions and the output neurons have linear ones.

[2]We are not criticizing this approach, we merely argue that at present it cannot be exploited for defining an appropriate measure for adaptability in artificial neural networks which are engineered for particular real world problems.

3.1 Real world problem class

Machine condition monitoring is an area of growing interest for manufacturing and operation. In order to reduce maintenance costs it is necessary to detect mechanical failures using a monitoring system. Since malfunctions often produce extraordinary vibrations which could be detected by a human supervisor acoustic monitoring is a widely used technique. Especially for methods like non-parametric spectral estimation and time-frequency-analysis there is a need for pattern recognition methods which are able to handle high dimensional feature vectors and model the underlying data structure.

We consider the problem of monitoring the tyre pressure of cars using impact sound spectra. The system has to cope with variations of several parameters like speed, tyre type and street surface. It has been shown (Gayko and Goerick 1996) that this system is nonlinear and that no generation and propagation model exists. Therefore, ANNs are employed for the analysis. During the preprocessing step a spectral estimation is performed on the sampled and recorded data. The method of averaged periodograms is used to enhance the estimation accuracy. The components of the power density spectrum are used as input patterns $\boldsymbol{x}$ for the neural estimators.

The parameters of the environment like speed, tyre type and street surface are represented in the environment state vector $\boldsymbol{a}$. This vector consists of components which can either be observable or not. We assume that the input vector has a cardinal scale of measurement whereas the environment state-vector may consist of inputs with different scales of measurement. Both problems (artificial and real world) can be modelled as regression tasks, where an estimation of the target value $\hat{y}$ is calculated using the input vector $\boldsymbol{x}$. This configuration is shown in Figure 4 (a).

A comparison of a fully-connected MLP with an ANN which has a problem dependent structure has been carried out by Gayko, Lohmann, Voss, Sendhoff, and Zamzow (1997). Especially, due to the high input dimension, the need for optimizing the ANN's structure was identified. The EANN approach resulted in a specialized neural network with fewer connections, however the results were confined to one particular environment state-vector. Thus, a generalization (which we termed second order in the last section) between different environment state-vectors was not included. In the framework of this paper, the set of different environment state-vectors define a problem class and the design of an EANN approach to optimize networks which can switch between different environments is the main aim of the proposed approach. In particular we concentrate on one parameter of the environment state-vector, the tyre type. The aim is to optimize a network which can rapidly adapt to different set of tyres exploiting general information about this environment state parameter which has been accumulated during the evolution both in the structure and in the initial weight setting.

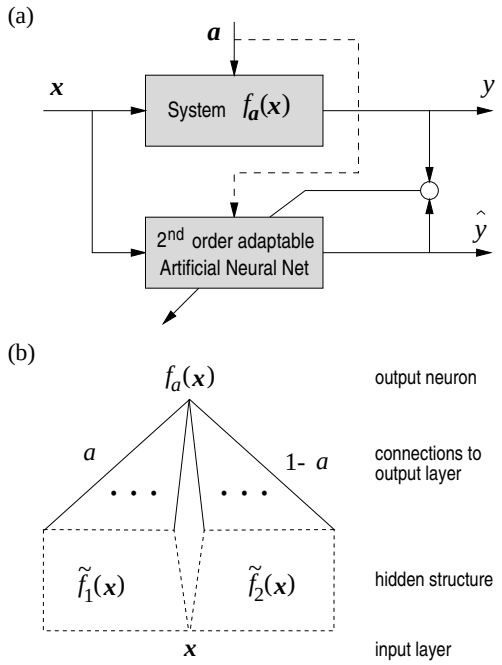

Figure 4: (a) A system characterized by $f_{\boldsymbol{a}}(\boldsymbol{x})$ with inputs $\boldsymbol{x}$, output $\boldsymbol{y}$ and environment state vector $\boldsymbol{a}$. Part (b) outlines a possible network structure tuned to a problem class, given by equation (1) and represented by the state parameter a.

3.2 Artificial problem class

To investigate different methods for developing second order adaptive ANNs, we create an artificial class of regression problems. Different sets of data are generated using a function $f_a(\boldsymbol{x})$ with the additional parameter a. Each fixed value of a defines one particular problem of the class. Of course, a is only used to generate the problems and to label them, but not as an additional hint for the solution. In this paper we employ the following function:

$$f_a(\boldsymbol{x}) = a\tilde{f}_1(\boldsymbol{x}) + (1-a)\tilde{f}_2(\boldsymbol{x}) \tag{1}$$

Therefore, in principle it is possible to organize the ANN in such a way that the commonness of the problems in the class can be exploited. For example, the structure given in Figure 4 (b) seems to be suitable for the problem class, since only the weights of the connections between the hidden and the output neurons have to be re-adjusted. It is likely that more suitable structures of the ANN exist, which additionally take the properties of the learning algorithm into account. In order to define the problem class (1) completely, the functions

$$\tilde{f}_1(\boldsymbol{x}) = \frac{1}{2+\sqrt{2}}\left[x_1 + \sqrt{x_2^2 + x_3^2} + \frac{\sin\left(\pi\frac{x_4 x_5}{2}\right)}{x_6+1}\right] \tag{2}$$

and

$$\tilde{f}_2(\boldsymbol{x}) = \frac{2}{4+\sqrt{2}}\left[\sqrt{\frac{1}{2}x_2 x_3} + \cos\left(\pi\frac{\sqrt{x_4^2+x_5^2}}{4}\right) + (x_6 - x_7)^2\right] \tag{3}$$

with $\boldsymbol{x} \in \mathbb{R}^8$ are used. The functions share the inputs $x_2, \ldots, x_6$; in addition $\tilde{f}_1(\boldsymbol{x})$ depends on x_1 and $\tilde{f}_2(\boldsymbol{x})$ on x_7, the component x_0 is not used. Furthermore, $0 \leq \tilde{f}_{1,2}(\boldsymbol{x}) \leq 1$ holds and different combinations of the input variables are realized.

For each problem four data sets are generated, containing 300 data points each. The input variables $0 \leq x_i \leq 1$ are chosen at random and noise (signal to noise ratio: $20dB$) is added to the target values $f_a(\boldsymbol{x})$.

4 Algorithms for finding second order adaptive ANNs

In this section, we present different approaches for developing ANNs and compare their adaptability towards different problems. First, we generate specialized ANNs and demonstrate their lack of adaptability towards related problems. In order to promote second order generalization, we discuss two different approaches in detail: the ensemble method and the generation method, see Section 2 and Figure 3 (a) and (b). We present results from experiments, using the artificial problem class proposed in the last section. In Section 5, we apply the methods to the real world problem of tyre pressure estimation, see Section 3.1.

As it is not the purpose of this paper to design the best possible ANN, but to introduce the idea of second order adaptivity and to discuss different approaches to solve this problem, we restrict ourselves to a standard evolutionary algorithm. However, it is straightforward to combine our ideas on problem class optimization with more elaborate evolutionary algorithms for structure optimization.

4.1 General proceeding for optimizing ANNs

In the following, we briefly summarize the main components of the different evolutionary algorithms we applied for structure optimization. The particular choice of the fitness function, the mutation operators and the selection scheme are given in Sections 4.2, 4.3 and 4.5.

The ANNs, i. e. the connections as well as the weights, are encoded in the individual's genome using a direct encoding scheme. Therefore, the structure and the weights can be optimized at the same time and in principle it is possible to use the knowledge gained from learning for weight modifications (see e. g. the approaches proposed by Lohmann (1993), Braun and Zagorski (1994), Yao and Liu (1997)).

The outline of the evolutionary algorithm is close to Evolutionary Programming (EP). Variations of the individuals

only take place due to different mutation operators and no recombination among different individuals is used. In most cases the parents of the succeeding generation are chosen using the EP-tournament-selection (Fogel 1995). In Section 4.5 a non-elitistic selection method is needed, therefore, we applied a (μ, λ)-selection as well as a non-elitistic tournament selection.

The fitness ϕ of an individual mainly depends on the error after a period of learning. The RPROP-Algorithm (Riedmiller 1994) is employed to minimize the mean squared error $e = \frac{1}{P}\sum_{p=1}^{P}(\hat{y}_p - y_p)^2$. Here, P denotes the number of data points presented and $\hat{y}_p$ and y_p denote the ANN's output and the target value for each data point p. To avoid overfitting and to increase generalization towards different data stemming from the same system (first order generalization), three data sets are employed. For the adjustment of the weights the training set is used. The error e_{test}, relevant for the fitness evaluation, is calculated on the test set, using the weight configuration with a minimum error on the validation set during learning. The maximum number of training cycles is limited and additionally, an early stopping mechanism is employed (Prechelt 1998).

Different methods exist to enforce sparse ANNs, i. e. ANNs with a low number of neurons or connections. Some implicit ways of doing so are to limit the CPU time or the number of cycles used for learning (Lohmann 1993) or to choose the mutation operators such that sparser ANNs are created with a higher probability (Braun and Zagorski 1994; Yao and Liu 1997). In this paper, we prefer an explicit approach which enforces ANNs of almost the same size. In this way, the different results are easier to compare. In the fitness function a penalty term $\phi_{\mathrm{pen.}}$ is added to the error criterion after learning. If the individual's size exceeds a given threshold, $\phi_{\mathrm{pen.}}$ becomes positive, otherwise it vanishes. Since the value of $\phi_{\mathrm{pen.}}$ depends on how much the threshold is exceeded, slight violations can be tolerated, if they correspond to a decrease of the error after learning. The threshold is chosen such that it is still possible to solve the problem with a high accuracy.

The mutation operators utilized in this paper are similar to those discussed in Braun and Zagorski (1994). Small, normally distributed modifications of the weights with a mean of zero are performed using the operator $\hat{M}_{\mathrm{jogWeights}}$. Additionally, two groups of operators are used for mutating connections and neurons. The simpler operators $\hat{M}_{\mathrm{ConSimple}}$ and $\hat{M}_{\mathrm{UnitSimple}}$ add and delete single connections and hidden neurons randomly without taking the weights into consideration. In order to minimize the impact of deletion, neurons with a lower number of connections are deleted with a higher probability. The operators $\hat{M}_{\mathrm{ConSoft}}$ and $\hat{M}_{\mathrm{UnitGain}}$ for adding and deleting connections and neurons take the values of the weights into account. The significance of connections and neurons is estimated by the absolute value of the weight and the gain. The gain was proposed by Goerick (1998) as a measure of how much each neuron contributes to the approximation task. Thus, connections with smaller weights and neurons with smaller gain are deleted with a higher probability.

a	$\sigma^2\ (10^{-2})$	$e_{\mathrm{gen.}}^{8-6-1}\ (10^{-4})$ (best / average)	$e_{0,\mathrm{gen.}}\ (10^{-4})$
0.0	1.04	1.61 / 2.70	1.97
0.2	0.80	1.30 / 2.47	1.31
0.5	0.83	1.12 / 1.37	1.42
0.8	1.27	2.28 / 3.41	2.11
1.0	1.84	2.60 / 4.22	2.35

Table 1: Error $e_{0,\mathrm{gen.}}$ on the generalization data set of the specialized ANNs, which is used in this paper to normalize all results. For comparison, the variance σ^2 of the target data and the error $e_{\mathrm{gen.}}^{8-6-1}$ of a fully connected 8-6-1-ANN (with about four times more weights than the optimized ANNs) after training is given.

4.2 Adaptability of specialized ANNs

Firstly, we perform structure optimization in the "classical" way, i. e. to find an ANN with a minimum approximation error and a minimum number of connections for one specific problem; we will term the resulting networks specialized ANNs. In our case, each problem is described by a data set resulting from one particular value of a (a=0.0, 0.2, 0.5, 0.8, 1.0). For each value of a the results are averaged over three trails, i. e. three different ANNs which resulted from different optimization runs.

The generalization error $e_{0,\mathrm{gen.}}$ of the specialized ANNs, calculated on a data set not used during learning, as well as the variance σ^2 of the target data[3] and the error $e_{\mathrm{gen.}}^{8-6-1}$ of a fully connected ANN with 6 hidden neurons, containing about four times more weights than the optimized ANNs (both values are only given for comparison) are shown in Table 1.

The results show that the different problems have slightly different complexities, measured in terms of σ^2 as well as in terms of the minimum error $e_{0,\mathrm{gen.}}$. In the following, we will normalize all errors by $e_{0,\mathrm{gen.}}$, assuming this to be the minimal possible error. For values of a not given in Table 1 we interpolate linearly. In only a few cases, the specialized networks are outperformed by other networks after adaptation, i. e. normalized errors smaller than one are reached, see Figure 5 (a), a=0.2. These cases are likely to be the result of stochastic fluctuations and not of any additional hints (Al-Mashouq and Reed 1991) contained in other problems from the same class.

Next, we investigate the ability of these specialized ANNs to adapt to other problems from the same class, which we termed second order generalization in the previous sections. Therefore, we *re*train the specialized ANNs towards a prob-

[3]The variance σ^2 equals the error one would make, if the mean value of the target data is used for the prediction of the next data point (for all data points).

lem with a different value a for 2000 cycles [4]. Again, we use the training data set for learning and the test data set for calculating the error of the ANN with the smallest error on the validation data set.

The results in Figure 5 (a) show that the minimum error is reached for the problems close to the one which the ANNs are specialized for. In the other cases, the error clearly increases as a function of a, i. e. the larger the difference in a, the larger the performance decrease. The very weak adaptability of the ANNs specialized for $a = 0$ and $a = 1$ is due to the structure of the problem class, equation (1). In these cases, the ANNs are specialized only to $\tilde{f}_1(\boldsymbol{x})$ or $\tilde{f}_2(\boldsymbol{x})$, respectively, therefore, no information about the structure of the other function is available during the evolutionary optimization.

These results do not allow a statement on the structure, i. e. the neurons and their connections, of the ANN, as the different errors after adaptation can strongly be influenced by specialized weight initialization, caused by the optimization subject to one particular problem. In order to circumvent this problem, we performed with the resulting ANNs a second series of experiments whereby only the adaptability of the structure was examined. As it is well known, that the capacity to learn as well as the learning speed of an ANN strongly depends on the initialization of the weights, it is suggested to initialize the weights at random with small values (LeCun, Bottou, Orr, and Müller 1998). We re-initialized the weights of the specialized ANNs and repeated the adaptation towards different problems. The results, averaged over 50 different weight initializations and over the three different ANN's structures per specialized ANN are given in Figure 5 (b).

The qualitative characteristics of the results are similar to the ones without re-initialization of the ANN's weights, Figure 5 (a). This is particularly obvious for the limiting cases $a = 0$ and $a = 1$, but can also be observed for medium values of a. This indicates that the results presented in Figure 5 (a) are not mainly caused by an inappropriate configuration of initial weights, but are truly a property of the ANN's structure.

The increase of the absolute error can be disregarded. It reflects the fact that ANN learning often gets stuck in local minima (Iyer and Rhinehart 1999), depending on the initial weights. This problem becomes more serious, the lower the number of adjustable parameters is. In this respect the optimized weight setting seems to be better (in most cases) than the random weight initialization even for problems other than the specialized value of a.

Summarizing, the ANNs, specialized to one problem, have not the capability to adapt to all the problems of the class; therefore, they are *truly* specialized solutions and do not provide second order generalization. This property is influenced by their connectivity as well as their weights. Although this result might not be particularly surprising (why

[4]This number has proved to be sufficient to adapt to different problems of this particular problem class.

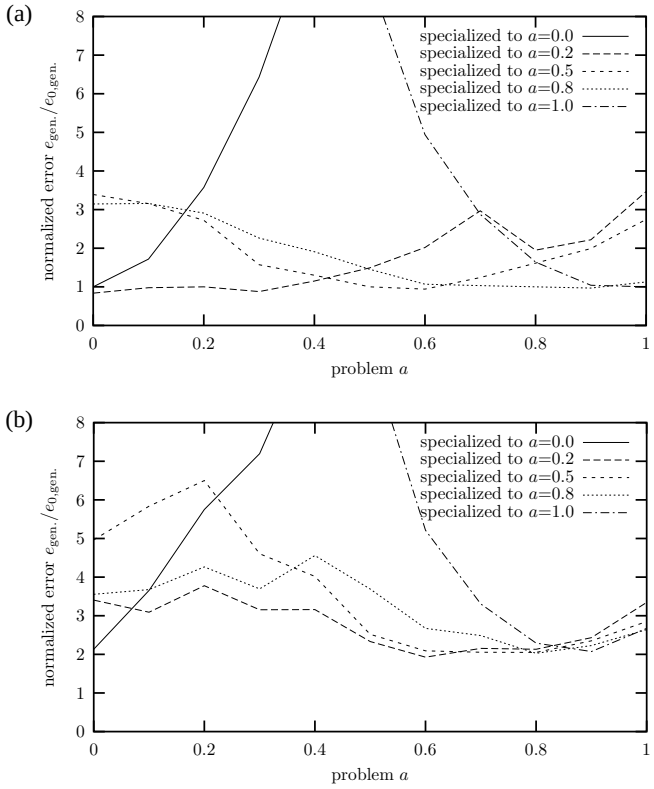

Figure 5: Adaptation of specialized ANNs to different problems (a) without and (b) with re-initialization of the ANN's weights before the adaptation process.

should the network generalize from one problem to another if it has only been optimized and trained for one specific problem), it highlights the need for additional measures which have to be taken during optimization. In Sections 4.3 and 4.5, we will discuss two very different approaches to incorporate information about the problem class during the developmental process to find the connections as well as suitable weights of second order adaptive ANNs.

4.3 Ensemble method – Evaluating the adaptability

We have shown in Section 4.2 that ANNs resulting from a standard algorithm for structure optimization do not provide second order adaptability, i. e. the ability to adapt to other, but related problems, than that, the ANNs are optimized for. To overcome this drawback, more than one problem has to be taken into account during the development of the ANNs. Two different approaches of doing so are the generation method, introduced in Section 4.5, and the ensemble method, which is investigated in this section.

The basic idea of the ensemble method (EnseM) is to perform an evolutionary optimization on a set of problems, belonging to the same class. The fitness ϕ is chosen as an estimation of the adaptability of the network to different problems. This is achieved by multiple learning of different problems and combining the resulting errors, see Section 2 and Figure 3 (a). Therefore, second order generalization is

achieved in a similar way as first order generalization in neural network learning by using several data sets. In contrast to the latter, here the data sets are not from one problem but from different problems belonging to one class.

4.3.1 Objective function for measuring adaptability

The variable $e_{\text{test}}^{(\text{adap})}$ denotes the lack of second order adaptability of an ANN. The superscript and subscipt indicate that it is the error after adaptation to different problems, evaluated on the test data set. In order to estimate $e_{\text{test}}^{(\text{adap})}$, the ANN is sequentially adapted to k different problems from the problem class, see Figure 3 (a).

Let $e_{\text{test}}^{(i)}$ denote the error after adaptation towards problem i, evaluated on the test data set. The value $e_{\text{test}}^{(\text{adap})}$ as a function of $e_{\text{test}}^{(i)}$ should fulfil the following requirements:

- Monotonically increasing in $e_{\text{test}}^{(i)}$.
- Symmetric in $e_{\text{test}}^{(i)}$, if all problems are assumed to be equally important.
- For comparability with the errors of a certain problem, $e_{\text{test}}^{(\text{adap})}$ should have the same dimension as $e_{\text{test}}^{(i)}$.
- If all the $e_{\text{test}}^{(i)}$ are equal, then $e_{\text{test}}^{(\text{adap})} = e_{\text{test}}^{(i)}$.

One function satisfying these requirements is

$$e_{\text{test}}^{(\text{adap})} = \sqrt[n]{\frac{1}{k}\sum_{\text{i}=1}^{k}\left(e_{\text{test}}^{(i)}\right)^n} \quad \text{with} \quad n \epsilon \mathbb{N}^{>0} \qquad (4)$$

In equation (4), n influences the importance of the different errors $e_{\text{test}}^{(i)}$; the larger n, the larger is the influence of the largest error and the larger is the selective pressure to decrease it. At the same time, the selective pressure to further decrease the smaller errors is reduced. One disadvantage of (4) becomes apparent, if the complexities, i. e. the minimal possible errors of the different problems, denoted by $e_{0,\text{test}}^{(i)}$, are very different. In this case, the influence of the most complex problem is strongly overestimated. This difficulty can be overcome by normalizing all the errors to the minimal possible error:

$$e_{\text{test}}^{(i)} \longrightarrow \frac{e_{\text{test}}^{(i)}}{e_{0,\text{test}}^{(i)}}. \qquad (5)$$

In this case one has to keep in mind that $e_{\text{test}}^{(\text{adap})}$ does not have the dimension of an error any more, but becomes dimensionless with one as the lower bound.

In our experiments, we employ equation (4) with $n = 2$ without normalization. The fitness ϕ results from combining $e_{\text{test}}^{(\text{adap})}$ with the penalty term $\phi_{\text{pen.}}$ to favour sparse structures:

$$\phi = e_{\text{test}}^{(\text{adap})} + \phi_{\text{pen.}} \qquad (6)$$

4.3.2 Optimizing the ANN's weights – Lamarckism vs. Darwinism

The importance of optimizing the ANN's weights together with the structure of the ANN is well known, see also Section 4.2. For developing specialized ANNs, it has turned out to be helpful to follow the Lamarckian scheme of inheritance, i. e. to exploit the knowledge gained from learning to modify the weights systematicly, in contrast to the random mutative weight modifications. On the other hand, this scheme has proved to be destabilizing in case of dealing with different problems and changing environments (Sasaki and Tokoro 1998). In order to study this situation in more detail, we investigate three different algorithms, which differ in the way how the initial weights are chosen as well as in the kind of inheritance.

EnseM$_1$: The initial weights are neither encoded nor inherited; they are chosen at random for each individuum in each generation.

EnseM$_2$: To avoid the noise resulting from the random choice of the initial weights in EnseM$_1$, the weights are encoded. Following the Darwinian scheme of inheritance, the weights are only modified by mutations using the operator $\hat{M}_{\text{jogWeights}}$.

EnseM$_3$: To take advantage of the knowledge gained during the adaptation period, a heuristic is introduced, called *averaged Lamarckian inheritance* The idea is to analyze similarities and differences of the weights resulting from the adaptation to the different problems i. If one particular weight turns out to have similar values after adaptation independent of i, this value seems to be a convenient starting point for learning all problems of the problem class. Otherwise, if the weights turn out to be very different after learning, especially with different signs, a small value should be chosen as the starting point to increase plasticity. One straightforward way to realize this idea is to encode back the arithmetic mean of the weights resulting from learning the different problems.

Since in these cases the weights do not seem to contain significant information about the importance of neurons or connections any more, only the simple mutations $\hat{M}_{\text{ConSimple}}$ and $\hat{M}_{\text{UnitSimple}}$ are used for structure modifications.

4.4 Numerical results of the Ensemble Method

In this section, we analyze the ANNs resulting from the algorithms EnseM$_1$, EnseM$_2$ and EnseM$_3$, using the artificial problem class, see Section 3.2. The lack of second order adaptability $e_{\text{test}}^{(\text{adap})}$ is evaluated in all algorithms by the help of $k = 5$ different problems ($a = 0, 0.2, 0.5, 0.8, 1$); additionally EnseM$_3$ is performed with only $k = 3$ problems ($a = 0, 0.5, 1$). All the results are averaged over three different trails.

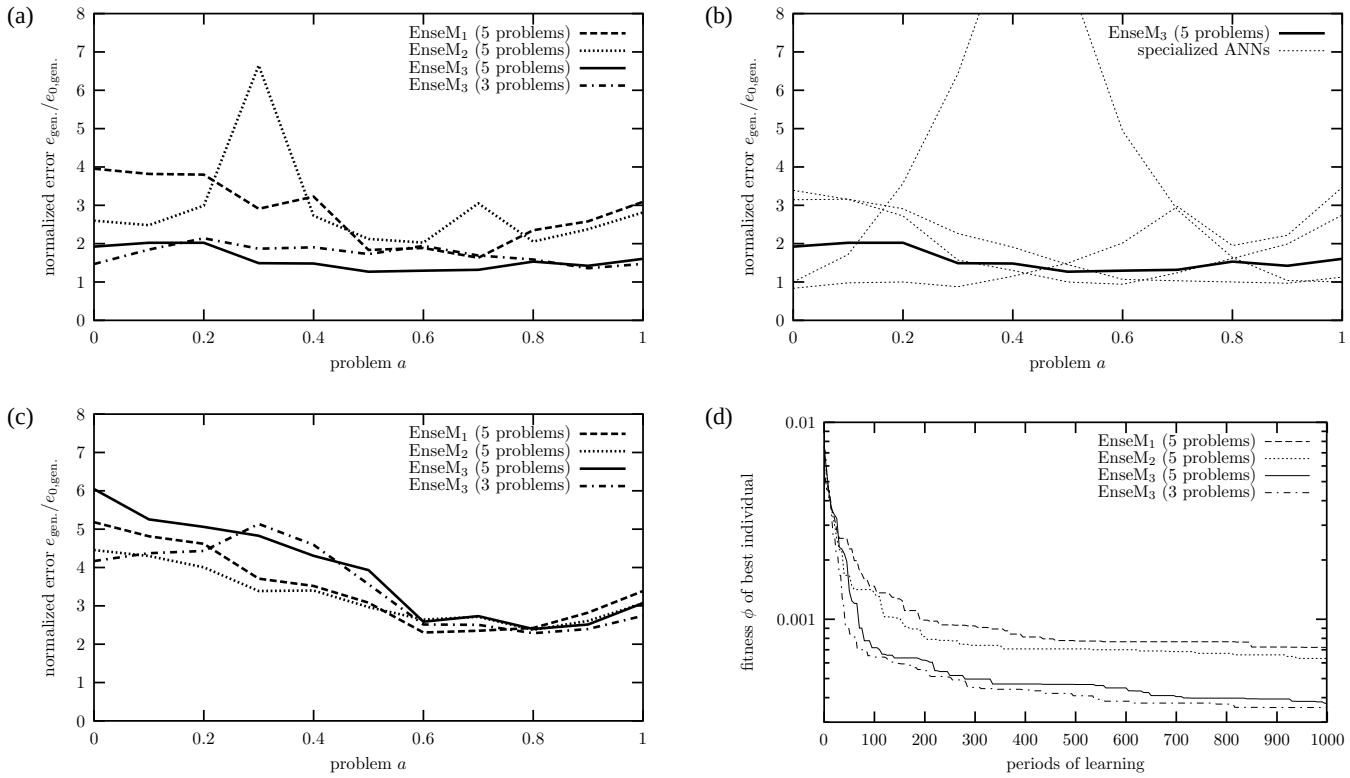

Figure 6: Results of the ensemble method: (a) error $e_{\text{gen.}}$ of the ANNs resulting from EnseM$_1$, EnseM$_2$ and EnseM$_3$ after adapting to the different problems of the problem class; (b) comparison of the adaptability of the best adaptive ANN (EnseM$_3$, 5 problems) with the specialized ANNs (see Section 4.2); (c) the same as (a), but after re-initialization of the ANN's weights; (d) Fitness ϕ of the best individual of each generation of EnseM$_1$, EnseM$_2$ and EnseM$_3$. In order to take the different computational costs of the fitness evaluation into account, the time is measured in terms of periods of learning, defined as the number of adaptation processes per generation and per offspring.

The temporal development of the fitness ϕ, which is mainly determined by the ability to adapt, is given in Figure 6 (d). It is obvious that the error decreases fastest compared to the other two algorithms when using EnseM$_3$; this result indicates again the importance to choose the initial weights in accordance to the ANNs structure and vice-versa. Especially, the approach to combine the knowledge gained from learning different problems to improve the ANNs weights (averaged Lamarckian inheritance) seems to be beneficial and more effective than a simple mutative search.

We next investigate the adaptability and second order generalization ability of the ANNs, which result from the ensemble method. The results are given in Figure 6 (a). As indicated by the development of ϕ, the adaptability of the ANNs resulting from EnseM$_3$ is superior compared to the results of the algorithms EnseM$_1$ and EnseM$_2$. In particular EnseM$_3$ shows the highest generalization ability towards unseen problems of the problem class. Additionally, in contrast to EnseM$_1$ and EnseM$_2$, the error $e_{\text{gen.}}$ is similar for all problems. It should be noted that the high second order generalization ability of EnseM$_3$ is even preserved, if only three instead of five problems are considered during the evolutionary optimization.

In Figure 6 (b) the adaptability of the ANNs resulting from the ensemble method are compared with the specialized ANNs. As we expected, the specialized solutions perform better on their specific problem a_{spec} and on the ones which are very close (usually problem $a_{\text{spec}} \pm 0.2$) with the exception of the extremal values. However, for the other problems, the situation is reversed and the second order adaptive ANNs, especially the ANNs resulting from EnseM$_3$, show better performance.

The adaptability of the ANNs generated by EnseM$_1$ and EnseM$_2$ is better than the adaptability of the specialists for extreme values of a like $a \simeq 0$ or $a \simeq 1$, but worse than specialists for medium values of a. The latter fact results from the comparatively high adaptability of the specialists for medium values of a due to the structure of the problem class, equation (1). However, in real world applications usually no order parameter like a is known and no constructive approach is available to decide, which specialists cover the main characteristics of the problem class and will therefore show a high degree of adaptability. Thus, EnseM$_1$ and EnseM$_2$ also represent systematic approaches to generate adaptive structures. Nevertheless, EnseM$_3$ is to be preferred.

As in Section 4.2, we study the adaptability of the structures of the ANNs by means of averaging over different initial weights; the results are given in Figure 6 (c). The adaptability to the different problems turns out to be comparable for all problems (at least compared with the specialized ANNs), but the absolute value is weak, compared with Figure 6 (a).

Like for the specialized ANNs, this result indicates that the property of being specialized or not is mainly determined by the ANN's structure, and that the initial weights have to be chosen *suitable for each structure*.

As pointed out before, the main benefit of EnseM$_3$ over the other two algorithms is the superior optimization of the initial weights due to the introduced heuristic. This fact is confirmed by these results, because when disregarding the weights, all the structures resulting from the different approaches perform similarly. Moreover, the performance of the structure resulting from EnseM$_3$ is slightly worse compared to the other structures. This can be understood from the design of the evolutionary algorithm. Since EnseM$_1$ and EnseM$_2$ modify the weights by re-initialization or by undirected mutations, there is a need for the individuals to develop robustness against the choice of the initial weights. Since in EnseM$_3$ the weights are chosen very specifically depending on the structure and the problems, no robustness towards fluctuations in the weight initialization is promoted during optimization.

4.5 Generation method – Averaging over time

Another way to develop second order adaptive ANNs is based on evolution in changing environments and will be called *generation method* (GeneM) in this work. The idea is to select the individuals in generation t for problem i_t and in the next generation for a different problem i_{t+1}. In contrast to the ensemble method, the ability to adapt to different problems is not evaluated each generation, but averaged over successive generations, as it is necessary to develop adaptability to survive in the population. It is experimentally shown that evolution of the initial weights under changing environments yield individuals with a higher ability to learn and to adapt to their environment (Nolfi, Miglino, and Parisi 1994; Nolfi and Parisi 1997; Sasaki and Tokoro 1997; Sasaki and Tokoro 1998).

Due to the evaluation of the ANNs for different problems in each generation, some principle difficulties arise. The first question concerns the strategy to choose the problems i_t, the individuals have to adapt to. It is important not to choose the same problem for succeeding fitness evaluations of one individual or its offsprings ($i_t \neq i_{t-1}$), otherwise the pressure for second order adaptation would be reduced. Additionally, i_t can be chosen separately for every individual or one i_t for the whole generation. As some problems i_t and some combinations of i_t and i_{t+1} are easier to adapt to (in case of the artificial problem class, it might be easier for an individual which is selected with respect to $a = 0.2$ to adapt to $a = 0.5$ than to $a = 1.0$), we prefer the latter method in order to give all the individuals the same chance to survive. Additionally, we chose i_t at random, but different to i_{t-1} to avoid any systematic asymmetry in the sequence of the problems. Additionally, due to the changing environment, either no elitistic selection scheme can be used or the fitness of the parents has to be re-evaluated in each generation. Therefore, in contrast to the ensemble method the fitness will not decrease monotonously, but oscillate.

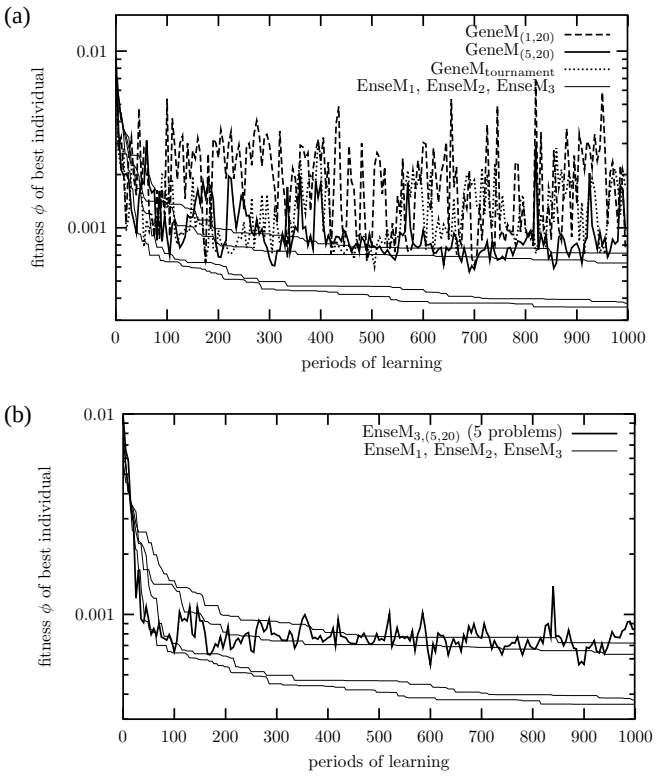

Figure 7: (a) Comparison of the adaptability of the ANNs, resulting from the generation method (thick lines) as well as from the ensemble method (thin lines, see Figure 6). (b) EnseM$_{3,(5,20)}$ is an example for the ensemble method, using a non-elitistic (5,20)-selection.

In our experiments, we employed the (μ, λ)-selection as well as a non-elitistic tournament-selection, whereby the fitness function is similar to the one from Section 4.2. In all algorithms, 20 offsprings are generated and adaptation takes place for 500 cycles. Modifications of the initial weights are only carried out by means of mutation, using the operator $\hat{M}_{\mathrm{jogWeights}}$, because it is not obvious, how to take advantage of the weight modifications during adaptation to one single problem. To compare the generation method with the results of the ensemble method, the error after adaptation to different problems ϕ, using equation (4) and the penalty term $\phi_{\mathrm{pen.}}$ as in Section 4.3, is calculated during the evolution and given in Figure 7 (a).

The ability to adapt to the problems of the problem class strongly oscillates due to the non-elitistic selection scheme. The mean of these oscillations (measured starting from generation 500) is $\bar{\phi} = 18.3 \times 10^{-4}$ for the (1,20)-selection scheme, $\bar{\phi} = 8.4 \times 10^{-4}$ for the (5,20)-selection (similar to Nolfi and Parisi (1997)) and $\bar{\phi} = 13.2 \times 10^{-4}$ for tournament-selection. They are worse than the results for EnseM$_1$ ($\phi = 7.2 \times 10^{-4}$) and EnseM$_2$ ($\phi = 6.3 \times 10^{-4}$). The fact that some of the oscillations of the adaptability ϕ are lower than EnseM$_1$ and EnseM$_2$ cannot be exploited from the technical point of view, since during the evolutionary process usually only the individual's fitness is known, which is not directly correlated with

ϕ. However, all of these results are outperformed by EnseM_3, which therefore has turned out to be the most applicable strategy. The remarkable result is that by an averaged Lamarckian scheme of inheritance it is possible to generate robustness against changing environments (in contrast to other investigations (Sasaki and Tokoro 1997; Sasaki and Tokoro 1998)) which is better than in an Darwinian approach.

In order to better understand the difficulties of using the generation method, EnseM_3 is utilized, but using a (5,20)-selection: $\text{Ense}\text{M}_{3,(5,20)}$, see Figure 7 (b). The magnitude of the oscillations is lower due to the explicit (and thus it seems more efficient) averaging over the adaptability towards different problems; the remaining oscillations stem from losing good individuals due to the non-elitistic selection scheme. This is also the reason for the saturation of ϕ starting in generation 100. In case of $\text{Ense}\text{M}_{3,(5,20)}$ this problem can simply be overcome by increasing the selective pressure (Hansen, Gawelczyk, and Ostermeier 1995), but in case of the generation method, this would be counterproductive, as the possibility to average over different problems would decrease (e. g. compare $\text{Gene}\text{M}_{(1,20)}$ and $\text{Gene}\text{M}_{(5,20)}$). These problems of the generation method become the more serious, the more problems of the class are considered.

5 Real world problem – system state diagnosis

In this section, we apply the methods developed in Section 4 to the real world problem of tyre pressure estimation, which we introduced in Section 3.1. All the ANNs, presented in this section have about 60 weights, which has turned out to be sufficient to avoid a significant increase of the error (Gayko et al. 1997). Unfortunately, only data for two different tyres are available, therefrom it is not possible to test the generalization property to other problems than the ones, which are incorporated into the developmental process.

The specialized ANNs are not suitable for estimating the pressure for other types of tyres than the one, they are specialized for. The error increases 430% on average, which is slightly better than the variance of the target data (540%). Furthermore, the specialized ANNs do not have the capacity to *adapt* to another problem than the one, they are specialized for. This is shown by adapting the specialized ANNs to the different types of tyres. The mean squared errors, calculated on a data set not presented during the evolutionary optimization, are given in the first two rows of Table 2. The error turns out to be 71% higher on average, if the non-specialized network is adapted.

In Section 4, different methods of developing second order adaptive ANNs were introduced. For the case of the artificial problem class, EnseM_3 turned out to be most efficient approach to determine the structure and initial weights of such an ANN. Therefore, we apply this method to develop an adaptable ANN for the tyre pressure estimation, called $\text{ANN}_{\text{Ense}\text{M}_3}$. The errors after adaptation of this ANN to the two types of tyres are given in the third row of Table 2. After

ANN	**MSE**		**normalized MSE**	
	tyre F	**tyre M**	**tyre F**	**tyre M**
ANN_F	0.0368	0.0458	1	1.68
ANN_M	0.0645	0.0272	1.75	1
$\text{ANN}_{\text{Ense}\text{M}_3}$	0.0427	0.0349	1.16	1.28
ANN_F (structure only)	0.0553	0.0613	1.50	2.25
ANN_M (structure only)	0.0735	0.0524	2.00	1.93
$\text{ANN}_{\text{Ense}\text{M}_3}$ (structure only)	0.0607	0.0544	1.65	2.00

Table 2: Mean squared error of the tyre pressure estimation of tyres of type M and F by different ANNs. Here, ANN_F and ANN_M denote ANNs, which are specialized to the tyres F and M, respectively, and $\text{ANN}_{\text{Ense}\text{M}_3}$ is an adaptive ANN, generated by the algorithm EnseM_3. In the last two columns, the errors are normalized to the value, which the specialized ANN has reached.

adaptation of $\text{ANN}_{\text{Ense}\text{M}_3}$ to the different tyres, the error is about 22% higher compared to the specialized ANNs. However, compared to the 71% increase of the specialized networks, this represents a considerable improvement. Whether the performance decrease compared to the specialized solution is acceptable has to be determined case by case.

Again, the investigation of the ANN's structure by means of averaging over different initial weight configurations demonstrates, that specialization and adaptability is a property of the structure itself. This can be seen by comparing the errors of $\text{ANN}_{\text{Ense}\text{M}_3}$ with the errors of the specialized structures $\text{ANN}_{\text{F,M}}$ after re-initialization of the weights (last three rows of Table 2). Since the results are qualitatively the same as without averaging over initial weight configurations, the structure itself has the properties described above. As before, the increase of the absolute errors is due to the problems with random weight initialization, see Section 4.2.

Summarizing, EnseM_3 has the ability to develop ANNs, that can adapt to different types of tyres better than specialized ANNs, which are designed by standard methods. This ability of the ANNs is influenced by a suitable choice of the structure of the connections and supported by an appropriate choice of the initial weights. Unfortunately, due to the limited number of data it was not possible to determine the second order generalization property with respect to other types of tyres.

6 Discussion and outlook

In this paper, we have focused on structure optimization of artificial neural networks (ANNs) for efficiently solving related problems, denoted as a class of problems. With reference to the generalization property of neural networks between different problems, we called this property second order generalization or second order adaptive networks.

Such a second order generalization of neural networks is desirable due to several reasons. Firstly, robust networks are needed for changing environments, which was demonstrated

in this paper by means of the application of networks to system state diagnosis for variable environment vectors, e. g. different tyre types. Secondly, the time consuming evolutionary optimization of ANNs will be more efficient if the resulting networks can be employed for a variety of tasks and not just for one specific problem. Thus, to develop ANNs, which have the capacity to adapt to problems from a given class of problems, is of major interest. For the practitioner this last point is a simple matter of a cost/benefit analysis.

ANNs resulting from standard algorithms for structure optimization do not have this second order generalization property. This is experimentally shown, using an artificial class of regression problems as well as the real world problem class of tyre pressure estimation. This ANN's property is influenced by both the structure of the neurons and their connections and the weights. We have investigated two different methods of developing second order adaptive ANN, called ensemble method and generation method. The first one, which has turned out to be more effective, uses the capability of the ANN to adapt to different problems of the problem class as the ANN's fitness; the best performance of this evolutionary algorithm was reached, when the knowledge gained during the adaptation to different problems was averaged and coded back into the genome – an averaged Lamarckian inheritance. In this case, the results are similar to the specialized ANNs, but clearly better than the specialists after adaptation to another problem. This result shows that also with a Lamarckian inheritance it is possible to generate robustness against changing problems (in contrast to other investigations, see (Sasaki and Tokoro 1997; Sasaki and Tokoro 1998)).

In the generation method, the individual's fitness is evaluated using only one problem, which has changed from generation to generation. Our preliminary experiments highlighted some general problems of this method, resulting from a non-elitistic selection scheme and the limited ability of the algorithm to average over different problems. Thus, the results are worse compared to the ensemble method. One possible reason, why a similar approach was more successfully applied in literature (Nolfi and Parisi 1997; Sasaki and Tokoro 1997) might be their limitation to two problems only as well as the restriction to the search of initial weights, leaving the connectivity unchanged. One possible way to overcome the problems of the generation method could be to average the individual's fitness explicitly over more than one generation: $\phi = (1 - c) \cdot \phi_{\text{parent}} + c \cdot e_{\text{test}}^{(i_t)}$. Here, $1/c$ determines the lifetime of the information for averaging.

Additionally, we have pointed out the importance of the structure and the initial weights for the attributes of the ANN. The property of being specialized or adaptable is strongly determined by the structure of the network. In addition, the right choice of initial weights is of great importance for successful learning. The intrinsic interplay between weights and network structure, however, is not yet fully understood.

In order to complete the preliminary results presented in this paper, the influence of different mutation operators and different selection-schemes have to be investigated. Additionally, a class of problems has to be defined more closely and a way has to be found to find representative problems of these classes. With respect to the application we have to analyze the performance of the networks on problems, i. e. on environmental state vectors, which have not been used before during the optimization. However, as we pointed out in the introduction, the collection of data for real world problems is both time-consuming and expensive.

Acknowledgement

We would like to thank Christian Igel, Reinhard Lohmann, Heiko Voss and Thomas Zamzow for fruitful discussions and helpful ideas. Part of this work is supported by the BMBF grant number 01 IB 701 A0.

References

Al-Mashouq, K. A. and I. S. Reed (1991). Including hints in training neural nets. *Neural Computation 3*(3), 418–427.

Braun, H. and P. Zagorski (1994). ENZO-M – a hybrid approach for optimizing neural networks by evolution and learning. In Y. Davidor, H.-P. Schwefel, and R. Maenner (Eds.), *Proceedings of the third Int. Conference on Parallel Problem Solving from Nature*, Jerusalem, Israel, pp. 440–451.

Crutchfield, J. and K. Young (1989). Computation at the onset of chaos. In W. Zurek (Ed.), *Complexity, entropy and the physics of information*, Santa Fe Institute Studies in the Sciences of Complexity, pp. 223–269. Addison Wesley.

Fahlmann, S. and C. Lebiere (1990). The cascade correlation learning algorithm. In D. Touretzky (Ed.), *Advances in Neural Information Processing Systems*, Volume 2, pp. 524–532. Morgan Kaufmann.

Fogel, D. B. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks 2*, 183–192.

Gayko, J. E. and C. Goerick (1996). Artificial neural networks for tyre pressure estimation. In *Proceedings of the International Conference on Engineering Applications of Neural Networks (EANN) '96*, pp. 13–16.

Gayko, J. E., R. Lohmann, H. Voss, B. Sendhoff, and T. Zamzow (1997). Application of structure evolution to system state diagnosis. In *Proceedings of the International Conference on Engineering Applications of Neural Networks (EANN) '97*, pp. 233–236.

Goerick, C. (1998). Considerations of the gain spectrum. Technical Report 98-2, Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany.

Hansen, N., A. Gawelczyk, and A. Ostermeier (1995). Sizing the population with respect to the local progress in (1,λ)-evolution strategies – a theoretical analysis. In *IEEE International Conference on Evolutionary Computation Proceedings*, pp. 80–85.

Iyer, M. S. and R. R. Rhinehart (1999). A method to determine the required number of neural-network training repetitions. *IEEE Transactions on Neural Networks 10*(2), 427–432.

LeCun, Y., L. Bottou, G. B. Orr, and K.-R. Müller (1998). Efficient backprop. In G. B. Orr and K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade*, pp. 9–50. Springer-Verlag.

Lohmann, R. (1993). Structure evolution and incomplete induction. *Biological Cybernetics 69*, 319–326.

Nolfi, S., O. Miglino, and D. Parisi (1994). Phenotypic plasticity in evolving neural networks. In D. Gaussier and J. Nicoud (Eds.), *Proceedings of the First International Conference From Perception to Action*, pp. 146–157. IEEE Computer Society Press.

Nolfi, S. and D. Parisi (1997). Learning to adapt to changing environments in evolving neural networks. *Adaptive Behavior 5*(1), 75–98.

Prechelt, L. (1998). Early stopping – but when? In G. B. Orr and K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade*, Chapter 2, pp. 55–69. Springer-Verlag.

Reed, R. (1993). Pruning algorithms - a survey. *IEEE Transactions on neural networks 4*(5), 740–747.

Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons – from backpropagation to adaptive learning algorithms. *International Journal of Computer Standards and Interfaces 16*(5), 265–278.

Sasaki, T. and M. Tokoro (1997). Adaptation toward changing environments: Why Darwinian in nature? In P. Husbands and I. Harvey (Eds.), *Fourth European Conference on Artificial Life (ECAL97)*, pp. 145–153. MIT Press.

Sasaki, T. and M. Tokoro (1998). Adaptation under changing environments with various rates of inheritance of acquired characters: Comparison between Darwinian and Lamarckian evolution. In B. McKay, X. Yao, C. Newton, J.-H. Kim, and T. Furuhashi (Eds.), *Simulated Evolution and Learning*. Springer.

Sendhoff, B. and M. Kreutz (1999). A model for the dynamic interaction between evolution and learning. *Neural Processing Letters 10*(3), 181–193.

Stagge, P. (1998). Averaging efficiently in the presence of noise. In A. Eiben, T. Bäck, M. Schoenauer, and H. Schwefel (Eds.), *Parallel Problem Solving from Nature – PPSN V*, Lecture Notes in Computer Science 1498, pp. 188–197. Springer.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE 87*(9), 1423–1447.

Yao, X. and Y. Liu (1997). A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks 8*(3), 694–713.