Structure Optimization of Neural Networks for Evolutionary Design Optimization

Michael Hüsken, Yaochu Jin, Bernhard Sendhoff

2005

Preprint:

This is an accepted article published in Soft Computing Journal. The final authenticated version is available online at: https://doi.org/[DOI not available]

Soft Computing (In press)
Special Issue on Approximation and Learning in Evolutionary Computation.Guest Edited by Y. Jin et al

Published in: Soft Computing, 9(1), pp. 21-28, 2005

Structure Optimization of Neural Networks for Evolutionary Design Optimization

Michael Hüsken[1], Yaochu Jin[2], and Bernhard Sendhoff[2]

[1] Institut für Neuroinformatik, Ruhr-Universität Bochum,
44780 Bochum, Germany
[2] Honda Research Institute Europe
63073 Offenbach/Main, Germany

The date of receipt and acceptance will be inserted by the editor

Abstract We study the use of neural networks as approximate models for the fitness evaluation in evolutionary design optimization. To improve the quality of the neural network models, structure optimization of these networks is performed with respect to two different criteria: One is the commonly used approximation error with respect to all available data, and the other is the ability of the networks to learn different problems of a common class of problems fast and with high accuracy. Simulation results from turbine blade optimizations using the structurally optimized neural network models are presented to show that the performance of the models can be improved significantly through structure optimization.
Keywords: Design optimization, neural networks, evolutionary algorithms, fitness approximation

1 Introduction

In many real-world applications of evolutionary computation, fitness evaluations are highly time-consuming. One attempt to reduce the computation time is to replace the original fitness function, at least in part, by an approximate model with a much lower computational cost [10]. Such models are also known as meta-models or surrogates in optimization. In [9] a framework for evolutionary optimization using approximate models with application to design optimization has been proposed. In this framework, the approximate model is combined with the original fitness function to control the evolutionary process, i.e., to decide how often the approximate model should be used instead of the original fitness function, to ensure the convergence of the evolutionary algorithm to a correct optimum of the original problem and to reduce the computational expense as much as possible. The basic idea is that the higher the quality of the model is, the more often it should substitute the original fitness function. This has been termed as evolution control in [9], also known as model management in design optimization with approximate models [2].

In many applications fully connected feed-forward neural networks have been used as approximate models, although it is well known that the approximation quality and learning efficiency of neural networks strongly depend on their architecture. The learning capability of the neural networks becomes particularly important when online learning needs to be performed during optimization. In this paper, structural optimization of neural networks is carried out before they are employed for fitness evaluations in evolutionary design optimizations. Since structure optimization based on the approximation error does not explicitly deal with the learning capability of neural networks, a method for learning problem classes suggested in [6] is also investigated.

The quality of the neural network models is usually evaluated with the quadratic approximation error. However, the approximation task in the context of a meta-model is not the same as in the context of optimal prediction. For a meta-model a qualitative approximation is often sufficient, whereas prediction needs a minimal quantitative difference. The examples in Figure 1 (a) and (b) clarify what we mean by "qualitative". The approximation accuracy of the neural networks shown might be quite unsatisfying, but nevertheless, these approximate models are still able to lead an optimization algorithm to the correct minimum of the fitness function. In this sense, the quality of the model as a meta-model is sufficient, although the approximation error is high. Thus, it is worth considering other quality measures to evaluate neural networks that are used as surrogates in design optimization.

The remainder of the paper is organized as follows. In Section 2, the framework for model management in design optimization used in this paper is reviewed briefly. Several measures for the model quality are discussed in Section 3. Thereafter, two approaches to structure optimization of neural networks are presented in Sec-

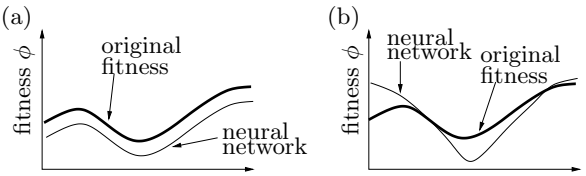

Fig. 1 Although the approximation errors of the neural network models are quite large, the optimization by means of the approximate models leads to the desired minimum of the fitness.

tion 4. The optimized networks are applied as approximate models in the design optimization of turbine blades in Section 5, where comparative studies are carried out to show the influence of the different strategies for neural network structure optimization on the design optimization outcome. A brief discussion and the conclusion of the paper is provided in Section 6.

2 The Framework for Model Management

It is not trivial to decide when to use the computationally efficient approximate model instead of the time-consuming original fitness function during the optimization. There are two basic issues that must be taken into account in applying approximate models in optimization. First, the optimization algorithm should converge to the global optimum of the original fitness function. As the empirical study in [8] indicates, the original fitness function should usually be used in over 50% of the fitness evaluations to guarantee the correct convergence of the evolutionary algorithm. Second, the approximate models should be used as often as possible to reduce the computation time.

To this end, a framework for model management in design optimization has been proposed in [9, 10]. The basic idea of the framework is that the higher the model quality, the more often the approximate models should be used. As shown in Figure 2, the evolutionary design optimization process is divided into succeeding *control cycles* consisting of a sequence of ζ generations. In the figure, individuals evaluated using the original fitness function are denoted by a shaded rectangle, whereas the individuals evaluated using the approximate model are represented by an unfilled rectangle. In the first η generations of the control cycle, the individuals are evaluated using the original fitness function, in the remaining generations, fitness evaluations are performed using the approximate model. The individuals in these η generations are denoted as *controlled individuals*. During the first η generations, the model output is compared with the original fitness function to evaluate the model quality to adapt the frequency value η. Moreover, the generated data of the original fitness function can be used to train the neural network models during the design optimization.

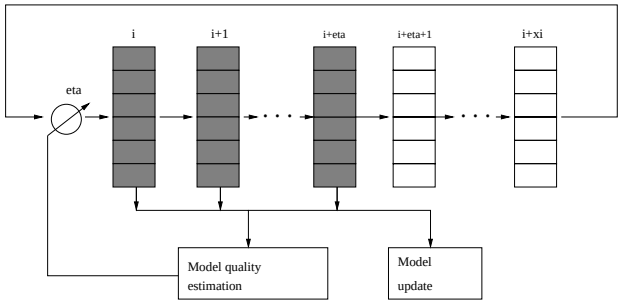

Fig. 2 A framework for model management in evolutionary design optimization. The gray and white boxes indicate individuals that are evaluated by means of the original fitness function or the approximate model, respectively. Each column of boxes indicate one generation.

3 Measures for the Quality of Approximate Models

As previously mentioned one of the main issues in the design and use of approximate models is their quality. However, quality is not inevitably equivalent to a close quantitative approximation of the fitness function, but should reflect the purpose in the design of the model, i.e., to ensure the selection of the best individuals in terms of the original fitness function, see Figure 1. In the following, we define different measures based on these considerations. Some of these measures rely on the particular selection scheme of evolution strategies. However, the basic idea can straightforwardly be applied to any kind of evolutionary algorithms.

3.1 Definition of Quality Measures

The most popular measure for model quality is the mean squared difference between the individual's original fitness function $\phi^{(\mathrm{orig.})}$ and the output of the approximate model $\phi^{(\mathrm{model})}$

$$E^{(\mathrm{mse})} = \frac{1}{n}\sum_{j=1}^{n}\left(\phi_j^{(\mathrm{model})} - \phi_j^{(\mathrm{orig.})}\right)^2 \quad . \qquad (1)$$

Here, the mean squared difference is averaged over n different individuals taken into account for the estimation of the quality measure, e.g., the $n = \lambda$ offspring individuals in one generation or the $n = \eta\,\lambda$ controlled individuals in one control cycle. When using (1) during training, additional mechanisms to avoid overfitting should be applied.

Generally speaking, a model with good approximation quality ensures the correct evaluation and consequently the correct selection of the individuals. However, from the evolutionary optimization point of view, only the correct selection is of importance. In the following, we define a number of measures that focus primarily on the correct model-based selection and not on the approximation accuracy. The exact definitions of the first two

measures depend on the selection method. Although we only give expressions for the case of the (μ, λ)-selection with $\lambda \geq 2\mu$, which is of particular relevance in our design optimization algorithm, it is in principle possible to extend the ideas and expressions to other selection schemes.

The first measure we suggest is based on the number of individuals that have been selected correctly using the approximate model:

$$\rho^{(\text{sel.})} = \frac{\xi - \langle \xi \rangle}{\mu - \langle \xi \rangle} \quad , \tag{2}$$

where ξ $(0 \leq \xi \leq \mu)$ is the number of correctly selected individuals, i.e., the number of individuals that would have also been selected if the original fitness function had been used for fitness evaluation. The expectation

$$\begin{aligned} \langle \xi \rangle &= \sum_{m=0}^{\mu} m \frac{\binom{\mu}{m}\binom{\lambda-\mu}{\mu-m}}{\binom{\lambda}{\mu}} \\ &= \frac{\mu^2}{\lambda} \quad . \end{aligned} \tag{3}$$

of ξ in case of random selections is used as a normalization in (2). It can be seen that if all μ parent individuals are selected correctly, the measure reaches its maximum of $\rho^{(\text{sel.})} = 1$, and that negative values indicate that the selection based on the approximate model is worse than a random selection.

The measure $\rho^{(\text{sel.})}$ only evaluates the absolute number of correctly selected individuals. However, in case of $\rho^{(\text{sel.})} < 1$ the measure does not indicate, whether the $(\mu+1)$-th or the worst offspring individual has been selected, which may have significant influence on the evolution process. Therefore, the measure $\rho^{(\text{sel.})}$ is extended to include the rank of the selected individuals, calculated based on the original fitness function. A model is assumed to be good, if the rank of the selected individuals based on the approximate model is above-average according to the rank based on the original fitness function. The definition of the extended measure $\rho^{(\sim\text{sel.})}$ is as follows: The approximate model gets a grade of $\lambda - m$, if the m-th best individual based on the original fitness function is selected. Thus, the quality of the approximate model can be indicated by summing up the grades of the selected individuals, which is denoted by π. It is obvious that π reaches its maximum, if all μ individuals are selected correctly:

$$\begin{aligned} \pi^{(\text{max.})} &= \sum_{m=1}^{\mu} (\lambda - m) \\ &= \mu \left(\lambda - \frac{\mu+1}{2} \right) \quad . \end{aligned} \tag{4}$$

In analogy to (2) the measure $\rho^{(\sim\text{sel.})}$ is defined by transforming π linearly, using the maximum $\pi^{(\text{max.})}$ as well as the expectation $\langle \pi \rangle = \frac{\mu\lambda}{2}$ for the case of a purely random selection:

$$\rho^{(\sim\text{sel.})} = \frac{\pi - \langle \pi \rangle}{\pi^{(\text{max.})} - \langle \pi \rangle} \quad . \tag{5}$$

Besides these two problem-dependent measures for evaluating the quality of the approximate model, two established measures — the rank correlation and the (continuous) correlation — partially fit the requirements formulated above. The rank correlation [13], given by

$$\rho^{(\text{rank})} = 1 - \frac{6 \sum_{l=0}^{\lambda} d_l^2}{\lambda(\lambda^2 - 1)} \quad , \tag{6}$$

is a measure for the monotonic relation between the ranks of two variables. In our case, d_l is the difference between the ranks of the l-th offspring individual based on the original fitness function and on the approximate model. The range of $\rho^{(\text{rank})}$ is the interval $[-1; 1]$. The higher the value of $\rho^{(\text{rank})}$, the stronger the monotonic relation with a positive slope between the ranks of the two variables. In contrast to $\rho^{(\sim\text{sel.})}$, the rank correlation does not only take the ranking of the selected individuals, but also the ranks of all individuals into account. This is more than needed for the model-based selection, however, it gives a good estimation of the ability of the model to distinguish between good and bad individuals, which is the basis for a correct model-based selection.

Another possibility to quantify the idea that the approximate model should ensure correct selection, but not necessarily reproduce the correct fitness values, is given by the (continuous) correlation between the approximate model and the original fitness function:

$$\rho^{(\text{corr.})} = \frac{\frac{1}{n} \sum_{j=1}^{n} \left(\phi_j^{(\text{model})} - \bar{\phi}^{(\text{model})} \right) \left(\phi_j^{(\text{orig.})} - \bar{\phi}^{(\text{orig.})} \right)}{\sigma^{(\text{modell})} \, \sigma^{(\text{orig.})}} \quad . \tag{7}$$

Here, $\bar{\phi}^{(\text{model})}$ and $\bar{\phi}^{(\text{orig.})}$ are the mean values and $\sigma^{(\text{modell})}$ and $\sigma^{(\text{orig.})}$ the standard deviations of the approximate model output and original fitness function, respectively. The properties of the correlation is related to both the rank based measures introduced above and the mean squared error. It is not a measure for the difference between model output and original fitness, but evaluates a monotonic relation between them. In addition, the range of this measure is known and therefore $\rho^{(\text{corr.})}$ is easier to evaluate than $E^{(\text{mse})}$. Besides, $\rho^{(\text{corr.})}$ is differentiable, which allows to use gradient-based methods for the adaptation of the model.

4 Evolutionary Optimization of Neural Networks

The performance of neural networks does not only depend on the choice of the weights, but also strongly

on the choice of the architecture or structure[1], i.e., the graph describing the number of neurons and the way the neurons are connected. In particular, the task of fast learning or learning with a small amount of data demands a suitable architecture [5]. Evolutionary structure optimization of the neural networks has proven to be a very efficient approach to choosing the architecture as well as the weights, refer to [15] for a survey. In principle, it is possible to embed the structure optimization of the approximate model into the design optimization algorithm. In this work, we perform structure optimization of the neural networks offline, i.e., before the neural networks are employed as meta-models in the design optimization algorithm. The data for this offline optimization stems from previous design optimization runs. Only the weights will be adapted in every single control cycle during the design optimization as outlined in Section 2. Further details about the structure optimization of the neural networks, in particular the incorporation of life-time learning are given in the following.

4.1 Accuracy-Based Structure Optimization

We employ a direct encoding scheme for the structure optimization of neural networks, i.e., every connection and the value of every weight of the networks are encoded in the individual's genome. Taking into account the characteristics of the structure of neural networks, specific mutation operators have been chosen. Single connections and neurons can be inserted or deleted. Weights are mutated by adding normally distributed random numbers. After mutation, τ iterations of gradient-based learning are introduced using iRprop$^+$ [7], an improved version of the Rprop-algorithm [12]. The purpose of the gradient learning is to fine tune the weights based on the mean squared error of the neural network. The modified weights are encoded back into the individual's genome after learning, following the effective Lamarckian paradigm [14]. Finally, we use EP-tournament-selection based on fitness values representing the mean squared error of the individuals after learning. To avoid overfitting we use early stopping during learning as well as different data sets for learning and for fitness evaluations.

A schematic illustration of the Lamarckian mechanism is shown in Figure 3. Note that the architecture a_j of the network encoded by the j-th individual does not change during life-time learning, but that the weights do; here $\boldsymbol{w}_j$ and $\boldsymbol{w}'_j$ denote the weights before and after learning, respectively. The variable $\mathcal{P}$ denotes the problem the networks should learn.

This kind of optimization searches for neural networks that represent the input-output mapping induced by a given set of data of the problem $\mathcal{P}$ with a minimum error, including the ability to generalize towards

[1] We will use the terms architecture and structure synonymously throughout the paper.

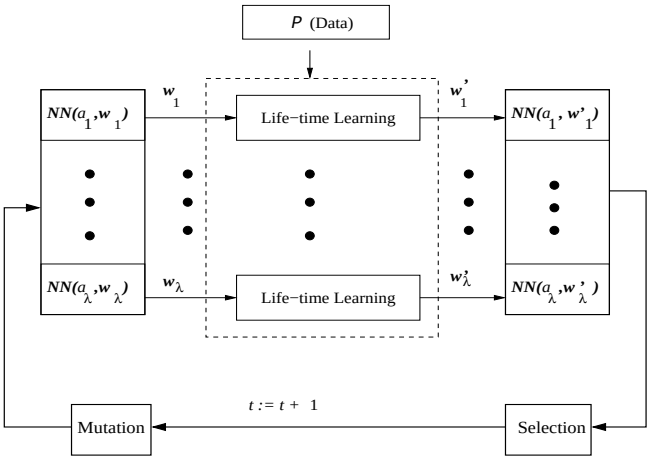

Fig. 3 The Lamarckian mechanism for evolutionary structure optimization of neural networks. The variable $NN(a_j, \boldsymbol{w}_j)$ stands for an individual encoding a neural network with architecture a_j and weight vector $\boldsymbol{w}_j$.

other data stemming from the same problem $\mathcal{P}$. Therefore, the result is one approximation model with one architecture and initial weight configuration for the whole fitness landscape, i.e. for the mapping from the design space into the performance space.

4.2 Structure Optimization for Problem Classes

As discussed in Section 1, the learning *capability* of neural networks can be as important as the pure performance, the approximation error. This is particularly the case when neural networks are learned online based on a small amount of data, like in the design optimization application where the data is taken from single control cycles. Here the neural networks serving as approximate models are trained at the end of each control cycle, as illustrated in Figure 2. It is intuitive, that the optimal models for the data from one control cycle to the next will share some common architectural aspects, since the population of the design optimization algorithm moves slowly compared to the duration of one control cycle. Furthermore, it can be seen that even different design optimization runs share common aspects.

In the last section, the different local approximation problems in each single control cycle were treated as one complex problem. The basic idea behind the approach outlined in this section is firstly to regard each single problem as unique and secondly to group these different problems in a problem class. Note that in general the definition of problem classes is difficult; no appropriate metric exists to define a proper notion of relatedness. In our application, we can circumvent an explicit definition by using the implicit one which we gain from the control framework outlined in Section 2. The approximation in each single control cycle is *one* problem, the group of all problems defines the class. The advantage of keeping

the problems separate is to give a clear definition to what we mean by the network's capacity to learn: How good is the network prepared by the structure optimization task to learn new problems from the class. Thus, if the network meets new areas of the fitness landscape, which is very likely to happen during optimization, the structure which reflects the common aspects of the problem class, allows the network to adapt to this new approximation problem fast and based only on few data. This type of generalization has been termed second order generalization in [4] because instead of generalizing between different data sets, the network has to cope with varying problems.

In [4], the authors investigate different structure optimization approaches in order to integrate common aspects of the problem class in the network's structure such that the network is well prepared for learning the different particular problems, i.e. the models in different control cycles. We extend and apply these methods to the domain of approximate modeling. Assume that data from ν control cycles are available for offline structure optimization of the neural networks. Instead of learning the whole problem $\mathcal{P}$ during life-time in one generation of the structure optimization, the problem is divided into ν subproblems $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_\nu$ according to the ν evolution control cycles. During the life-time learning of the j-th individual ($1 \leq j \leq \lambda$) the network represented by this individual is first trained on the data of the problem $\mathcal{P}_1$ for τ iterations, resulting in the weight vector $\boldsymbol{w}_{j,1}$ and the approximate error $E_{j,1}$. Thereafter, this network with weight configuration $\boldsymbol{w}_{j,1}$ is further trained on the problem $\mathcal{P}_2$, resulting in the weight vector $\boldsymbol{w}_{j,2}$ and the approximation error $E_{j,2}$. This process continues until the j-th network has been trained on all ν problems. A schematic illustration of the algorithm is shown in Figure 4.

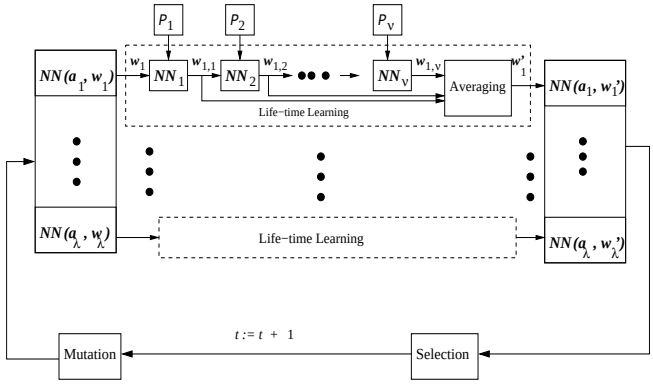

Fig. 4 Evolutionary structure optimization of neural networks for problem classes with averaged Lamarckian inheritance.

This kind of structure evolution is supported by means of the *averaged Lamarckian inheritance*, which is explained in the following. As learning deals with a number of different problems, some kind of average weights have to be coded back [6]. Since in the next generation these weights will be the starting configuration in the first control cycle, the influence of the learned weights $\boldsymbol{w}_{j,i}$ should decline with increasing index i of the control cycle:

$$\boldsymbol{w}'_j = \frac{1-\gamma}{1-\gamma^\nu} \sum_{i=1}^{\nu} \gamma^{i-1} \boldsymbol{w}_{j,i} \quad ; \quad 0 < \gamma < 1 \quad , \tag{8}$$

where γ is a coefficient that balances the influence of the different problems. The j-th individual's fitness is given by the average training error on the different problems:

$$\phi_j = \frac{1}{\nu} \sum_{i=1}^{\nu} E_{j,i} \quad . \tag{9}$$

There are two main features in the Lamarckian evolutionary optimization. First, the weights learned during life-time learning on a given problem (P) are encoded back to the chromosome. Second, the approximation error on all data after learning is assigned to the fitness.

The main difference between the structure optimization algorithms outlined in this and in the previous section is the target of the optimization. In the first one, all problems are grouped together into *one* approximation task is defined; target of the structure optimization is minimal approximation error on this problem. In the second set-up, all problems are kept separately defined by the partition of the design optimization into control cycles; target is the fast and efficient learning of each new problem based on a common architecture. This difference is reflected by the different algorithms illustrated in the Figures 3 and 4 and the different usages of the Lamarckian inheritance.

5 Application and Results

In this section, we apply approximate modeling to the domain of aerodynamic design optimization, in particular the optimization of transonic gas turbine blades. The performance of each blade is evaluated based on computational fluid dynamics (CFD) simulations. Navier-Stokes equations with the $(k-\epsilon)$ turbulence model are used in the two-dimensional CFD simulation [1].

Evolutionary algorithms have proven to be very promising for the optimization of these complex shapes [11]. However, thousands of performance evaluations are usually needed before a satisfactory solution can be obtained. Unfortunately, CFD simulations are very time-consuming. To cope with this difficulty, computationally efficient approximate models can be used to substitute for some of the CFD simulations.

5.1 Experimental Setting

In this work, a (μ, λ) evolution strategy with covariance matrix adaptation [3] without recombination is employed to minimize the normalized pressure loss Ω and the deviation of the outflow angle α at the trailing edge of a turbine blade from a desired angle of $\alpha_0 = 69.7°$. The fitness function of the evolution strategy is given by

$$\phi^{(\text{orig.})} = c_1 \, |\alpha - \alpha_0| + c_2 \, \Omega + P \quad , \tag{10}$$

where P is a penalty term from mechanical constraints; c_1 and c_2 are weighting factors for the pressure loss and the deviation of the outflow angle, respectively. They should be properly chosen so that the preference of the designer over the two terms can be reflected correctly. In this study, we set c_1 to 10 and c_2 to 1000. If the mechanical requirements, for example the minimal thickness of the blade, are not met, a very large penalty term is added to the fitness. Recall that we try to minimize the fitness in this application.

The sizes of parent and offspring populations are $\mu = 2$ and $\lambda = 11$, respectively. Since the length of a control cycle is $\zeta = 6$ generations, a maximum amount of $\lambda\,(\zeta - 1) = 55$ data points is available in each control cycle for learning. One feed-forward neural network is utilized for the approximation of each of the two performance indices Ω and α. We consider a two-dimensional optimization and the shape of the blade is represented with non-uniform rational B-splines with 26 control points. Therefore, there are 52 inputs, describing the shape of the blade, to the approximate models.

5.2 Optimization Results with Neural Networks

Three types of approximate neural network models are used in the design optimization and compared with respect to their ability to increase the performance of the design optimization. The model of the first type (ApxNN$^{(1)}$) use a fully connected architecture. The weights are initialized by means of offline learning, using a number of given training data collected in a comparable blade optimization trial (e.g., different initialization but the same number of control points of the spline and the same fitness function). The second type of network model (ApxNN$^{(2)}$) is obtained with regard to the approximation accuracy of all data points collected during the seven control cycles of a different evolutionary run. The corresponding structure optimization algorithm was outline in Section 4.1. The third type of model (ApxNN$^{(3)}$) results from using structure optimization with respect to its learning capability for $\nu = 7$ different problems stemming from the first control cycles of a different design optimization trial. The structure optimization algorithm for problem classes was outlined in Section 4.2. We use the value $\gamma = 0.5$ in the averaged Lamarckian inheritance, equation (8). For all types of models, after η generations of each control cycle in the design optimization the weights are adapted online for $\tau = 50$ iterations using iRprop$^+$.

Table 1 Best results achieved with the different types of approximate models.

	α	Ω	f
ApxNN$^{(1)}$	69.70°	0.061	61.59
ApxNN$^{(2)}$	69.70°	0.0542	54.20
ApxNN$^{(3)}$	69.68°	0.057	57.38

In each design optimization trial, a maximum number of 3000 evaluations of the CFD simulation was allowed, so that the design optimizations with the three different approximate models roughly need the same amount of computational costs. It is assumed that the computational time for fitness evaluations using the neural network model and for training the network is negligible compared to the CFD simulation, which is reasonable in our application. Since the evolution control frequency is adjusted during optimization, the number of fitness evaluations becomes different in different optimizations, although the number of CFD calls are the same. Table 1 summarizes the performances of the best blades obtained with the different kinds of approximate models, showing

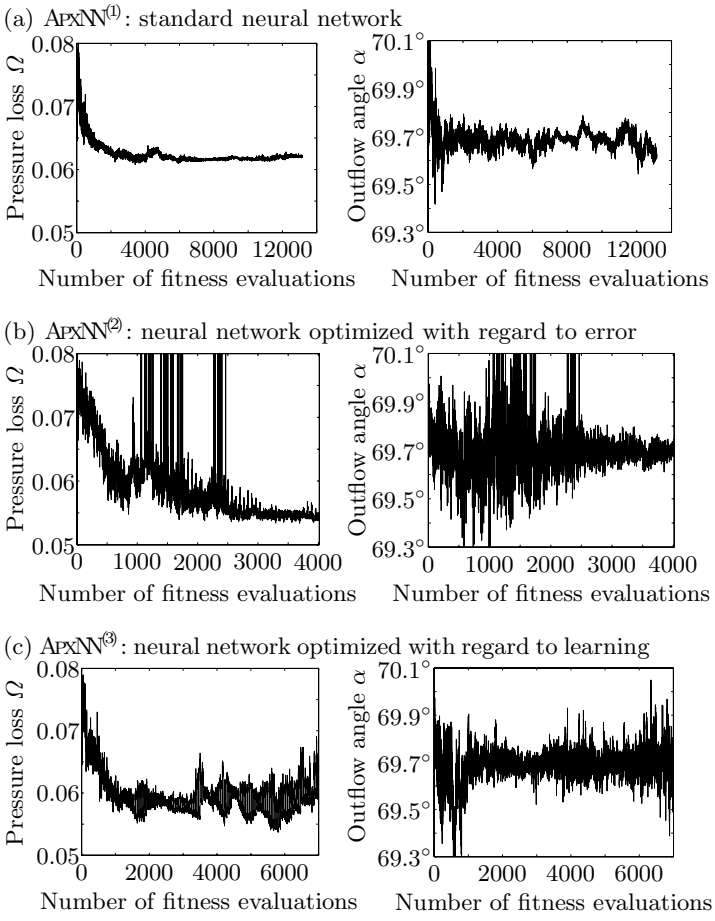

Fig. 5 Results of the blade optimization using three different types of approximate neural network models

that the use of a fully connected architecture seems to be the worst choice. Figure 5 depicts the evolution of the pressure loss and the outflow angle. In Figure 5 (b), we notice a number of extreme oscillations during the optimization. They result from individuals that are penalized, as the CFD simulation has not converged within a prescribed number of iterations, which is an undesired situation for the evolution. These oscillations are caused by the CFD simulation and are independent of the quality of the neural networks. Their occurrence varies between different runs and have been observed for all three set-ups. It is also noticed that the pressure loss goes up at the end of the optimization in Figure 5 (c). This is due to the fact that no elitism is introduced in the optimization. For a clearer comparison, Figure 6 depicts almost the same values, but averaged over all individuals in one generation; as only the controlled generations (i.e., the generations in which the CFD simulations are conducted) are considered, the horizontal axis approximately scales with the amount of computation.

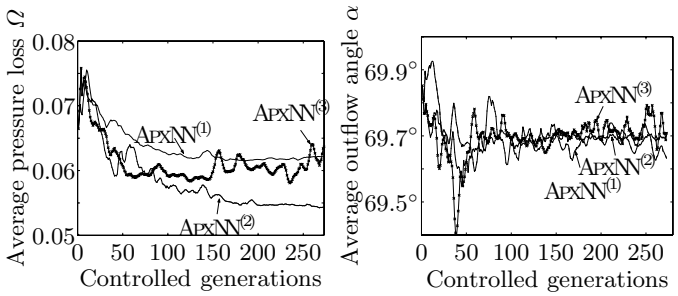

Fig. 6 Results normalized to the number of controlled generations (i.e., proportional to the number of CFD evaluations)

5.3 Comparison and Evaluation of the Measures

So far, the approximation error, i.e. the mean squared error, has been used as the quality criterion in our optimization. As discussed in Section 3, accuracy is not the only criterion for model quality when the model is used for fitness evaluations. For this reason, several measures have been suggested. To illustrate the relation between the different measures of the quality of the approximate model, we evaluate structurally optimized models for independent runs of the design optimization. In Figure 7 we compare $\rho^{(\sim\text{sel.})}$, the measure that seems to be of particular relevance for practical purposes, with the other four measures. First of all a mainly linear relation between the measures $\rho^{(\text{corr.})}$, $\rho^{(\text{rank})}$, $\rho^{(\text{sel.})}$ and $\rho^{(\sim\text{sel.})}$ becomes obvious, Figure 7 (a)-(c). Moreover, the relation between $\rho^{(\sim\text{sel.})}$ and $\rho^{(\text{rank})}$, Figure 7 (a), as well as $\rho^{(\sim\text{sel.})}$ and $\rho^{(\text{corr.})}$, Figure 7 (c) looks very similar, which is also emphasized by the high correlation between $\rho^{(\text{corr.})}$ and $\rho^{(\text{rank})}$ (not depicted). Compared with this result, the measure $\rho^{(\text{sel.})}$, Figure 7 (b), seems to be too

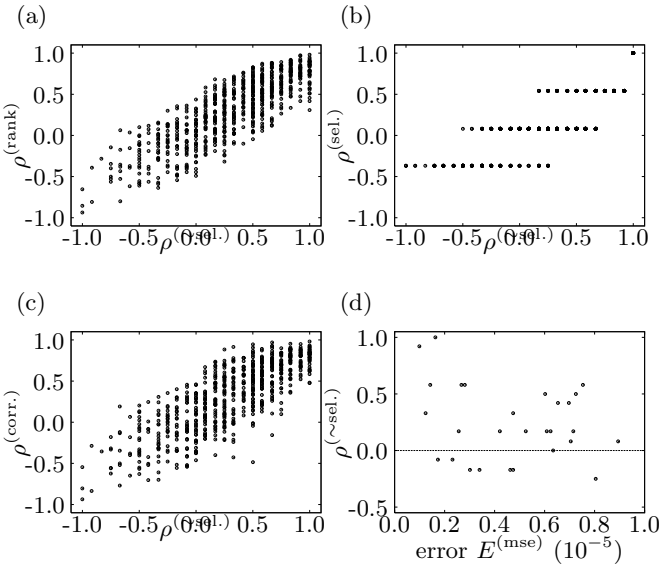

Fig. 7 Scatter plots to illustrate the relation between the different measures. Each circle corresponds to one model, evaluated based on the data of one generation.

coarse-grained to serve as a suitable basis for evaluating the different models.

As the range of $E^{(\text{mse})}$ strongly depends on the shapes of the blades, Figure 7 (d) is based only on the data from the same generation of the design optimization, evaluated with differently optimized models. For small values of $E^{(\text{mse})}$ the measure $\rho^{(\sim\text{sel.})}$ is decreasing with increasing $E^{(\text{mse})}$, for larger mean squared error $\rho^{(\sim\text{sel.})}$ is mainly fluctuating with a mean of approximately 0. In particular these strong fluctuations indicate, that $E^{(\text{mse})}$ is only weakly related to the ability to select the correct individuals. Due to the strong linear relation between $\rho^{(\sim\text{sel.})}$, $\rho^{(\text{sel.})}$, and $\rho^{(\text{rank})}$, this result can be carried over to the other measures.

One should keep in mind, that these results are based on models optimized with respect to $E^{(\text{mse})}$. Therefore they might change in case of a model selection based on other measures, preferably based on the relevant measure $\rho^{(\text{sel.})}$. The fact that $E^{(\text{mse})}$ is only weakly related to the important selection based measures, seems to indicate that the online training or at least the structure optimization algorithms should better employ one of these measures instead of $E^{(\text{mse})}$. The answer is "*maybe*"; in fact simply replacing $E^{(\text{mse})}$ by $\rho^{(\text{sel.})}$ does not work, since the information is too indirect, i.e. the relation between weight changes and changes in $\rho^{(\text{sel.})}$ is not obvious. On the other hand, one could imagine to employ a combination of $E^{(\text{mse})}$ and $\rho^{(\text{sel.})}$, e.g. to achieve a coarse approximation at first and in a second stage to evaluate the selection-based performance. Such an integrated approach to the evaluation of meta-models is currently under investigation.

6 Conclusions

Three types of neural network structures have been used for fitness approximation in evolutionary design optimization. It can be seen that both types of structurally optimized networks exhibit better performance than standard neural network models. This is of particular interest because the structure optimization of approximate models used for fitness evaluations in evolutionary computation is still often overlooked.

We introduced two different types of structure optimization algorithms, one which emphasizes the approximation quality of the optimized network and one which emphasizes the learning capability. As a result, the accuracy-based optimization outperformed the learning-based approach. This result was at first surprising for us, because the second set-up seems to fit more naturally into the problem decomposition offered by the design optimization algorithm. There are several possible explanations. One reason might be the fact that the data for the off-line structure optimization of the networks are generated with a fixed evolution control frequency, whereas the control frequency during online optimization is adapted according to the approximation error. In order to better compare both approaches the control frequency should be kept constant during the application as well. These experiments are currently under their way. An alternative explanation would be that the difference between the different problems is simply too small, therefore, the learning capability is not really "needed". Moreover, it will be interesting to investigate the performance of the neural networks when the optimization is carried out under different parameter settings. In that case, the third type of networks is expected to have better performance than the second type of networks.

Finally, we discussed several alternative measures for the quality of neural networks following the observation that networks with a limited accuracy can nevertheless serve as excellent meta-models during the design optimization, recall Figures 1 from the introduction. The surprising result was that the proposed selection-based measure, which is much closer to the task of the meta-model, is only weakly correlated to the standard approximation error. As a result, means have to be found to integrate both measures into a new approach for the evaluation of meta-models.

Acknowledgement

We would like to thank the BMBF, grant LOKI, number 01 IB 001 C, for their financial support of our research.

References

1. T. Arima, T. Sonoda, M. Shirotori, A. Tamura, and K. Kikuchi. A numerical investigation of transonic axial compressor rotor flow using a low-Reynolds number k-ϵ turbulence model. *Journal of Turbomachinery*, 121:44–58, 1999. Transactions of the ASME.
2. J. Dennis and V. Torczon. Managing approximate models in optimization. In N. Alexandrov and M. Hussani, editors, *Multidisciplinary design optimization: State-of-the-art*, pages 330–347. SIAM, 1997.
3. N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
4. M. Hüsken, J. E. Gayko, and B. Sendhoff. Optimization for problem classes – Neural networks that learn to learn. In Xin Yao and David B. Fogel, editors, *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (ECNN 2000)*, pages 98–109. IEEE Press, 2000.
5. M. Hüsken, C. Igel, and M. Toussaint. Task-dependent evolution of modularity in neural networks. *Connection Science*, 14(3):219–229, 2003.
6. M. Hüsken and B. Sendhoff. Evolutionary optimization for problem classes with lamarckian inheritance. In Soo-Young Lee, editor, *Seventh International Conference on Neural Information Processing (ICONIP 2000) – Proceedings*, volume 2, pages 897–902, 2000.
7. C. Igel and M. Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50:105–123, 2002.
8. Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimization with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 786–792. Morgan Kaufmann, 2000.
9. Y. Jin, M. Olhofer, and B. Sendhoff. Managing approximate models in evolutionary aerodynamic design optimization. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001)*, pages 592–599. IEEE Press, 2001.
10. Y. Jin and B. Sendhoff. Fitness approximation in evolutionary computation – A survey. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 1105–1112. Morgan Kaufmann, 2002.
11. M. Olhofer, T. Arima, T. Sonoda, and B. Sendhoff. Optimization of a stator blade used in a transonic compressor cascade with evolution strategies. In I. Parmee, editor, *Adaptive Computing in Design and Manufacture*, pages 45–54. Springer-Verlag, 2000.
12. M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 586–591. IEEE Press, 1993.
13. C. Spearman. The proof and measurement of association between two things. *American Journal of*

Psychology, 15:72–101, 1904.
14. D. Whitley, S. Gordon, and K. Mathias. Lamarckian evolution, the Baldwin effect and functional optimization. In Y. Davidor, H.-P. Schwefel, and R. Maenner, editors, *Parallel Problem Solving from Nature*, volume III, pages 6–15. Springer-Verlag, 1994.
15. X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.