Structure Optimization of Neural Networks for Evolutionary Design Optimization

Michael Hüsken, Yaochu Jin, Bernhard Sendhoff

2002

Proceedings of the Genetic and Evolutionary Computation Conference - Workshop, pp. 13-16, New York, July 2002

Structure Optimization of Neural Networks for Evolutionary Design Optimization

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

Yaochu Jin and Bernhard Sendhoff
Future Technology Research Division
Honda R&D Europe
63073 Offenbach/Main, Germany
yaochu.jin@de.hrdeu.com

Abstract

We study the use of neural networks (NN) as approximate models for fitness evaluation in evolutionary computation. To improve the quality of the NN models, structure optimization of these NNs is applied with respect to two different criteria: One is the commonly used approximation error, and the other is the ability of the NNs to learn different problems of a common class of problems. Simulation results from turbine blade optimization using the structurally optimized NN models are presented to show that the performance of the model can be improved significantly through structure optimization.

1 INTRODUCTION

In most applications of evolutionary computation the evaluation of the individual's fitness is the most time consuming component of the optimization. One attempt to reduce this time is to substitute the original fitness function—at least in some generations—by an approximate model with a much lower computational cost [7]. In [6], 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 to which proportion the approximate model and the original fitness function should be used to ensure the convergence of the evolutionary algorithm to a correct minimum of the original problem and to reduce the computational expense. The frequency at which the approximate model is used is determined by the estimated quality of the models. The higher the model quality is, the more often the approximate models are used instead of the original fitness function. In detail, the evolutionary process (in the following named *design evolution*) is divided into succeeding *control cycles* consisting of a sequence of β generations. In the first η generations of each control cycle, the individuals are evaluated by means of the original fitness function, in the remaining ones by means of the approximate model. During the first η generations, the model output is compared with the original fitness function to adapt the value of η; this procedure is denoted as *evolution control* [6].

Of course, the accuracy of the approximate model strongly determines the efficiency of this approach. As it is quite unlikely to find an accurate model for the whole fitness landscape, it seems to be more promising only to model the local vicinity of the actual population. Therefore, the genomes and the quality values of the individuals of the first η generations of the current control cycle are used to adapt the approximate model, starting from the approximate model in the last control cycle. We use feed forward neural networks (NNs) as approximate models and gradient-based learning for online adaptation of the NNs.

Since the performance of the NNs strongly depends on their architecture, we employ *structure optimization* of the NNs. In our context this is of particular importance, because the amount of available data is rather limited. During structure optimization, not only the conventional approximation error, but also the ability of learning have been used as criteria for the optimization.

The remainder of the paper is organized as follows. In Section 2, we introduce two approaches to structure optimization of NNs. The optimized NNs are applied as approximate models in the evolutionary design optimization of turbine blades in Section 3, where comparative studies are carried out to show the influence of the different strategies for NN structure optimization on the design optimization outcome. A brief discussion and the conclusion of the paper is provided in Section 4.

2 STRUCTURE OPTIMIZATION OF THE APPROXIMATE MODEL

The performance of NNs does not only depend on the choice of the weights, but also strongly on the choice of the architecture (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 available data demands a suitable architecture. *Evolutionary structure optimization* is a beneficial approach of choosing the architecture and the weights, refer to [10] for a survey.

In principle, it is possible to nest the structure optimization of the approximate model into the design optimization. However, we perform the structure optimization offline (i.e., prior to the design optimization) and only conduct the adaptation of the weights of the optimized NNs online in every single control cycle.

2.1 STRUCTURE OPTIMIZATION FOR NEURAL NETWORKS

A typical structure optimization algorithm, which is also employed in our investigations, can be outlined as follows: Each individual codes for the architecture and the weights of one NN by means of a direct encoding scheme (i.e., every connection and the value of every weight of the NN is encoded in the individual's genome). The mutation operators are chosen with respect to the characteristics of NNs: We employ the insertion and deletion of single connections and neurons, respectively, as well as normal distributed perturbations of all weights. After mutation, a period of gradient-based learning using iRprop$^+$ [5], an improved version of the efficient Rprop-algorithm [9], is introduced for an efficient fine tuning of the weights with respect to the mean squared error of the NN, calculated on a certain data set. After learning the modified weights are coded back into the individual's genome following the Lamarckian paradigm. Finally, we use EP-tournament-selection based on fitness values representing the error of the individuals on the problem at hand.

The standard optimization task is to structure a NN such that it represents the input-output mapping induced by a given set of data with a minimum error, including the ability to generalize towards other data stemming from the same process. As such kind of optimization tries to find one architecture and weight configuration suitable for all available data, the optimization would aim at an approximate model for the whole fitness landscape of the design evolution.

2.2 OPTIMIZATION FOR PROBLEM CLASSES

As stated in Section 1, we aim at optimizing a NN that allows for an accurate online adaptation of the approximate model, only based on the data collected during the first η generations of the current control cycle. The task of adapting the approximate model to the data from one particular control cycle is denoted as one *particular problem*. We assume that the problems in both, different control cycles and different evolutionary design optimization trials for the same design optimization task, are not completely different, but share some common aspects; they establish a *class of problems*. The architecture of the NN has to carry the information about the problem class and serves as some kind of regularization during learning. In [3], the authors investigated different structure optimization approaches in order to integrate common aspects of the problem class in the NN's structure such that the NN is well prepared for learning the different particular problems. We extend and apply these methods to the domain of approximate modelling.

The structure optimization is conducted based on data collected in ν control cycles of design evolution without approximate modelling or at least without a structure optimized approximate model. In comparison to the basic outline given in Section 2.1, the structure optimization targeting the learning ability slightly differs with respect to the learning phase and the fitness evaluation of the NN. Instead of learning with all data from all ν control cycles, every single control cycle is taken into account separately. For every control cycle learning is conducted for τ iterations using data collected in the first η generations of the cycle. The mean squared error of the trained NN is determined based on the remaining generations of the control cycle. The NN's fitness is given by the average mean squared error in the ν control cycles. In the first control cycle, learning starts from the genetically coded weight state, in the remaining cycles from the learning outcome in the previous control cycle.

This kind of structure evolution is supported by means of *averaged Lamarckian inheritance*. As learning deals with a number of different problems, some kind of average weights w have to be coded back [4]. Since these weights are the starting configuration in the first control cycle, the influence of the learned weights w_i should decline with increasing control cycle i. Note, that i indexes the number of control cycles, thus the history of learning of the network.

$$w = \frac{1-\gamma}{1-\gamma^{\nu}} \sum_{i=1}^{\nu} \gamma^{i-1} w_i \quad ; \quad 0 < \gamma < 1 \quad . \qquad (1)$$

3 APPLICATION AND RESULTS

In this section, we apply the approximate modelling 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 for the two-dimensional CFD simulation [1].

Evolutionary algorithms have proven to be very promising for the optimization of these complex shapes [8]. 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 the CFD simulations.

3.1 EXPERIMENTAL SETTINGS

In this work, a (μ, λ) evolution strategy with covariance matrix adaptation [2] 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

$$f = 10\,|\alpha - \alpha_0| + 1000\,\Omega + P \quad , \qquad (2)$$

where P is a penalty term from mechanical constraints. The weighting factors for the pressure loss and for the deviation of the outflow angle are chosen such that the influence of the two addends is balanced. 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 $\beta = 6$ generations, a maximum amount of $\lambda\,(\beta - 1) = 55$ data points is available in each control cycle for learning. One feed forward NN 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 to the NN models, describing the shape of the blade.

3.2 OPTIMIZATION RESULTS WITH NEURAL NETWORKS

Three types of approximate NN models are used in the design evolution and compared with respect to their ability to increase the performance of the design optimization. The models of the first type (ApxNN$^{(1)}$) use a fully connected architecture and 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 same fitness function). The second type of NN models (ApxNN$^{(2)}$) are obtained using evolutionary structure optimization, as discussed in Section 2.1, i.e., the NNs are optimized with regard to the approximation accuracy of all data points collected during the first seven control cycles of a different evolutionary run. The third type of models (ApxNN$^{(3)}$) result from using structure optimization with respect to its learning ability for $\nu = 7$ different problems stemming from the first control cycles of a different design optimization trial; we use the value $\gamma = 0.5$ in the averaged Lamarckian inheritance (1). For all types of models, after η generations of each control cycle in the design optimization online adaptation of the weights is performed for $\tau = 50$ iRprop$^+$ iterations.

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

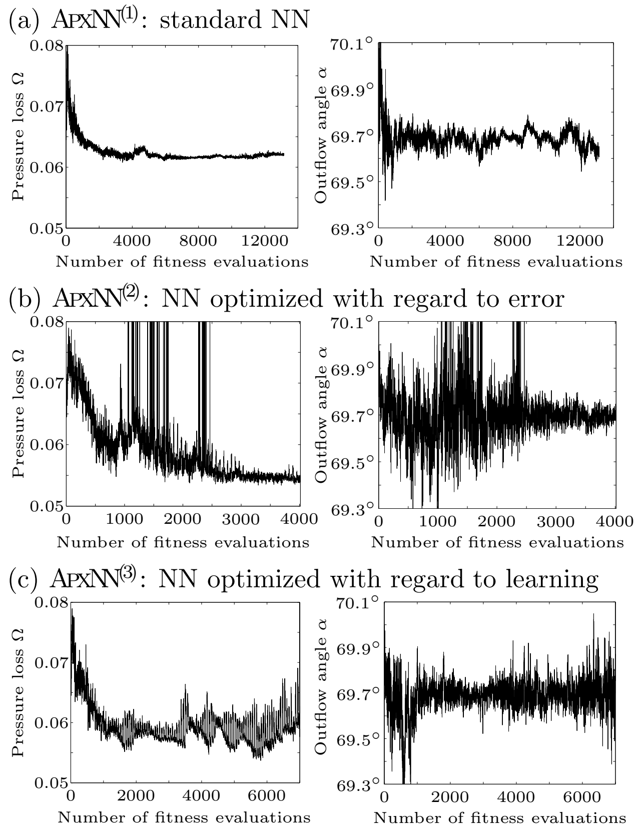

Figure 1: Results of the blade optimization using three different types of approximate neural network models

In each design optimization trial, a maximum number of 3000 calls of the CFD simulations was allowed, so that the design optimizations with the three different approximate models roughly need the same amount of computational costs. Table 1 summarizes the performances of the best blades obtained with the different kinds of approximate models, showing that the use of a fully connected architecture seems to be the worst choice. Figure 1 depicts the evolution of the pressure loss and the outflow angle. In Figure 1 (b), we notice a number of extreme oscillations during the optimization. These 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. For a clearer comparison, Figure 2 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.

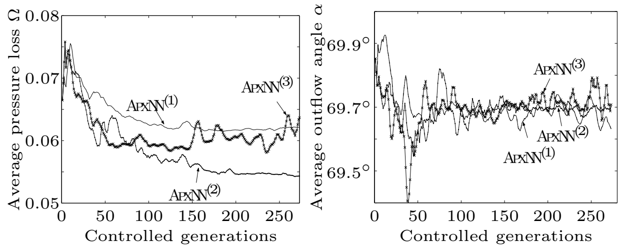

Figure 2: Results normalized to the number of controlled generations (i.e., proportional to the number of CFD evaluations)

4 CONCLUSIONS

Three neural network (NN) structures have been used for fitness approximation in evolutionary blade optimization. It can be seen that both types of structurally optimized NNs exhibit better performance than standard NN models. The NNs that have been structurally optimized with regard to the approximation accuracy with respect to all data at the same time achieved the best results. The reason for this is not clear so far, but we hope to figure it out by means of additional experiments. Moreover, it will be interesting to investigate the performance of the NNs when the optimization is carried out under different parameter settings. In that case, the third type of NNs is expected to have better performance than the second type of NNs.

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] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[3] Michael Hüsken, Jens E. Gayko, and Bernhard 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.

[4] Michael Hüsken and Bernhard 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.

[5] Christian Igel and Michael Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 2002. In press.

[6] Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. Managing approximate models in evolutionary aerodynamic design optimization. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001)*. IEEE Press, 2001.

[7] Yaochu Jin and Bernhard Sendhoff. Fitness approximation in evolutionary computation – A survey. In *Proceedings of Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 2002.

[8] 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, 2000.

[9] Martin Riedmiller and Heinrich 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.

[10] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.