Task-Dependent Evolution of Modularity in Neural Networks — A Quantitative Case Study

Michael Hüsken **Christian Igel** **Marc Toussaint**
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany
{michael.huesken,christian.igel,marc.toussaint}@neuroinformatik.ruhr-uni-bochum.de

Abstract

There exist many ideas and assumptions concerning the development and meaning of modularity in biological and technical neural systems. Nevertheless, this wide field is far from being understood; quantitative simulations and investigations are rare.

In our contribution, we empirically study the development of connectionist models in the context of the evolution of artificial neural networks for highly modular problems. We define two measures for the degree of modularity and monitor their values during the evolutionary process. We identify two different reasons for the development of modular structures: the modularity of the task is reflected by the modularity of the adapted structure and the demand for fast learning structures increases the selective pressure towards modularity. However, learning can also counterbalance some imperfection of the underlying structure.

1 Introduction

The performance of biological as well as technical neural systems (see [1] for an overview of neural computation) crucially depends on their architectures. For example, in case of an artificial feed-forward neural network (ANN) architecture may be defined as the underlying graph, which constitutes the number of neurons and the way these neurons are connected. Particularly one property of architectures, *modularity*, has raised a lot of interest among researchers dealing with biological and technical aspects of neural computation. It is obvious to emphasize modularity in neural systems because the vertebrate brain appears to be highly modular both in an anatomical and in a functional sense.

It is important to stress that there are different concepts of modules. "When a neuroscientist uses the word 'module,' s/he is usually referring to the fact that brains are structured, with cells, columns, layers, and regions which divide up the labor of information processing in various ways" [3]. A definition of modularity from the viewpoint of behavioral or cognitive science could be based on Fodor's work [4], as stated in [3]: "A module is a specialized, encapsulated mental organ that has evolved to handle specific information types of particular relevance to the species."

In general, modularity can be viewed as a fundamental design principle of complex systems. For example, the strategy to divide a complicated task into several smaller subproblems, which may be easier to solve than the whole task, has led to modular ANNs (e.g., see [10]), where an architecture consisting of *expert networks* is designed manually. Already in 1962, Simon [15] gave very elaborate arguments in favor of modularity. He claimed that the *development* of complex systems requires stable subsystems, which are combined in a hierarchical manner. Similarly, an engineer might argue that it is easier to design a complex system from smaller building blocks that have been tested in other systems and reliably perform a self-contained task. According to such arguments, modularity could be viewed not only as an outcome of system design but as a necessity for the development of highly complex systems.

In any case, the modularity of an architecture is a result—or a by-product—of a creation process, e.g., of the natural evolution in case of biological systems. In a technical framework, the role of natural evolution is often played by evolutionary algorithms (EAs): The problem of finding a suitable network architecture for a given task is often solved by means of evolutionary computation, see [19] for a survey. Based on arguments similar to the ones given above, EAs for topology optimization of ANNs have often been designed in such a way that modular topologies are favored. This is achieved by indirect encoding schemes, i.e., the adjacency matrix of the graph describing the ANN is not directly stored in the genome of an individual, but a non-trivial (e.g., grammar-based) genotype-phenotype-mapping is utilized to construct the ANN from a given genotype [11, 7, 6, 14, 13].

However, we believe that there are some fundamental, unanswered questions. Above all, we want to know: Under which circumstances do modular architectures of artificial (and—in future research—also biological) neural systems evolve? This basic question raises several others, for example: For what kind of problems are modular architectures favorable? How can modularity be quantified?

In this preliminary investigation, we test the hypothesis whether modular problems are solved better by modular ANNs than by non-modular ones. This claim appears natural and is often presumed in the literature. In this investigation, we utilize EAs to search for good ANNs for artificial modular problems. Our EAs have no explicit bias towards modular topologies; *if modular structures are beneficial, they will evolve.* If no increase in modularity is observed, the hypothesis has to be reconsidered. Measures that quantify modularity are proposed and employed to trace the evolution of modular topologies. Different problems are considered in order to test the task-dependency of the hypothesis. In particular, we want to verify that the demand for *fast* (gradient-based) *learning* increases the need for modularity.

We do not propose a general definition of modularity, but focus only on a specific character of problems and architectures, respectively: We call a problem modular if it can be solved by two non-interacting subsystems. In order to characterize the modularity of architectures we define two measures, which are related to the interaction between two parts of the network. Roughly, we count connections that link one part to the other and nodes that get only input from one part, respectively. As a major result, we find that the selective pressure towards the modularity of the architecture depends on the task. Especially when the speed of learning determines fitness, modularity is enforced.

In the next section we define a preliminary concept of modularity of ANNs and introduce two measures for its degree. In section 3 we present our simulations, and in the succeeding section, we discuss the results. The article ends with a conclusion and an outlook.

2 Modularity of Neural Networks

2.1 Topology of Neural Networks

In this investigation, we concentrate on artificial feed-forward neural networks (ANNs, see [1, 2]). The topology of an ANN can be defined by a directed (colored) graph, where the vertices correspond to the neurons and the edges indicate the flow of information [17]. The performance of an ANN depends on both its topology and the weights that are associated with its edges. Usually gradient-based optimization is applied for the adjustment of the weights, whereas the search for a suitable topology for a given task sets up a hard discrete optimization problem, which is tackled successfully by means of evolutionary algorithms (EAs) [19]. Evolution of connectionist models is not only carried out to improve the performance of neural networks for technical tasks, but also to investigate fundamental issues like the interaction between evolution and learning.

2.2 Measuring Modularity

Most people have an intuitive idea about what modular ANN structures are. Usually, they refer either to structural modularity, defining modules as highly connected areas that are only sparsely connected to the surrounding, or to functional modularity, defining modules as areas that fulfill a certain task [16]. However, giving a general mathematical definition and providing a measure for the degree of modularity capturing all concepts of modularity mentioned above and in section 1 seems to be a formidable task.

We restrict ourselves to a special, surely limited, definition of modularity to illustrate some basic concepts. We assume that a feed-forward neural network has to deal with a completely separable task. Here, separable means that the task can be solved by an input-output function f such that the output variables depend only on disjoint sets of input variables. As an example, we consider two output variables (y_1, y_2), where the input $(x_1, .., x_n)$ can be separated such that

$$\begin{aligned}(y_1, y_2) &= f(x_1, .., x_n) \\ &= \Big(f_1(x_1, .., x_m)\,,\ f_2(x_{m+1}, .., x_n)\Big) \quad . \quad (1)\end{aligned}$$

In this case, it does not seem to be disadvantageous to divide the whole network into two separated ones; such a system would be called highly modular. In this context, the degree of modularity is related to how strong the two parts are separated. In the following, we will give two different possible definitions of such a

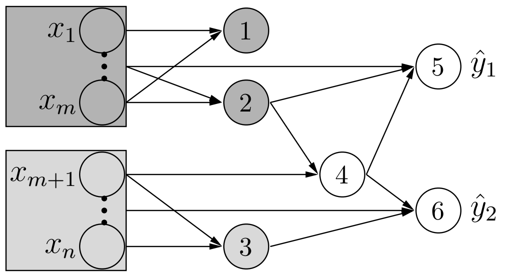

Figure 1: Sample topology to illustrate the two measures of the degree of modularity.

measure. Both measures do not only indicate whether an ANN is highly modular or not, but also quantify the degree of modularity. We assume that the task can be written in the form of equation (1) and that m is known.

Path-Count-Measure: The information is propagated from the input to the output through the connections. Therefore, the structure of the connections basically organizes the information flow through the network. Of course, it is also influenced by the weights and the shape of the neuron's activation functions, but we will neglect this for the definition of the measure of the degree of modularity.

The basic idea is to count all paths that connect an input neuron x_i with an output neuron whose target value, defined by task (1), depends on x_i. Let p_{ji} be the number of different paths from input x_i to output y_j. Then, the modularity is given by

$$\mathcal{M}^{(\text{paths})} = \left(\sum_{i=1}^{m} p_{1i} + \sum_{i=m+1}^{n} p_{2i} \right) / \sum_{j=1}^{2} \sum_{i=1}^{n} p_{ji} \quad . \quad (2)$$

This expression becomes 1 in case of a completely separated topology and is 0.5 for a random topology. The lower bound of this measure is 0. It is straight forward to extend definition (2) to (i) architectures with more than two output neurons, (ii) to problems consisting of more than two parts, and (iii) to problems where some input variables influence more than one part of the problem.

The topology in figure 1 may serve as an example. The number of paths from $x_1, \dots, x_m$ to the output $\hat{y}_1$ is five and to the output $\hat{y}_2$ is two. There exists one path from the inputs $x_{m+1}, \dots, x_n$ to $\hat{y}_1$ and there are four paths to $\hat{y}_2$. Therefore, there are 12 paths at all and the modularity measure becomes $\mathcal{M}^{(\text{paths})} = (5+4)/12 = 0.75$.

Neuron-Inputs-Measure: Another measure of the degree of modularity results from regarding each neuron as a basic information processing unit. The neurons are distinguished by their source of information: neurons that only get input, directly or indirectly, from either $x_1, \dots, x_m$ or $x_{m+1}, \dots, x_n$ are denoted as *pure* neurons. In the example in figure 1, the grey neurons (neuron 1, 2 and 3) are pure neurons. Neurons 4, 5 and 6 are *mixed* ones, as they receive input from both subsets of input neurons.

The second measure of the degree of modularity is defined as the rate of pure neurons, but disregarding all input neurons and all neurons that are not connected, directly or indirectly, to at least one output neuron:

$$\mathcal{M}^{(\text{neurons})} = \frac{n^{(\text{pure})}}{n^{(\text{pure})} + n^{(\text{mixed})}} \quad . \quad (3)$$

By definition, $\mathcal{M}^{(\text{neurons})}$ is between 0 and 1; a completely separated topology corresponds to a value of 1. In case of the example in figure 1, this measure becomes $\mathcal{M}^{(\text{neurons})} = 2/5 = 0.4$.

Both measures basically depict the information flow through the network, but a number of differences between them exist. The first measure focuses on input-output paths, the second one on the basic information processing units. Therefore, the latter one allows for identification and separation of modules more easily if modules are defined as those subgraphs with input from the same subproblem; this definition is related to the concept of strongly connected regions with a low degree of connectivity to the surrounding. Two such modules are indicated in figure 1 by the light and dark gray coloring. These modules are determined in a "bottom-up" fashion, a procedure that appears suitable to quantify to which extend a network can be regarded as a hierarchical assembly of modules. The latter requires that the measure is extended to deal with more than two modules and accounts for the number of different sub-problems that are fused in a mixed neuron.

The path-count-measure is less coarse than the neuron-inputs-measure (e.g., the number of different numerators in equation (3) is limited by the number of hidden and output neurons). This fact appears to make it less brittle, as experimentally shown in the following sections.

3 Experimental Framework

In this section, we present the framework of our simulations of the evolution of neural networks. During the simulations, we monitor the degree of modularity in the population. The results of these experiments are presented and discussed in section 4.

An interesting point in the experiments is that the optimization is conducted for two different tasks: One task is the optimization with respect to an *accurate model* (i.e., a neural network that solves the given problem as good as possible); this can be regarded as the classical optimization task, the search for a specialist for a given problem. The other task is to evolve a *fast learner*. Here, the aim is to find a network topology that can learn a given problem within a very short time. In section 4 we will see, that the definition of the task has a large impact on the development of modularity.[1]

3.1 Data Set

As discussed in section 2.2, we choose a completely separable classification problem. This problem is described by a data set that results from the two completely independent Boolean expressions

$$y_1 = (x_1 \text{ and } x_2) \operatorname{xor} x_3 \quad , \tag{4}$$
$$y_2 = (x_4 \text{ or } x_5) \operatorname{xor} x_6 \quad . \tag{5}$$

The data set contains all possible combinations of inputs (i.e., $2^6 = 64$ different patterns). To allow the processing by means of neural networks, "true" and "false" are encoded by "1" and "-1", respectively.

3.2 Neural Networks

We use feed-forward neural networks with six input and two linear output neurons. As the absolute value of the measure of modularity strongly depends on the number of hidden neurons and connections, we keep both of these values fixed during the experiments. Our experiments are performed with 20 hidden neurons with the sigmoidal activation function $\sigma(a) = a/(1 + |a|)$ and 60 connections. The resulting structures are large enough to solve the problem, but sparse enough to allow for the evolution of separated modules. However, experiments with slightly modified network sizes yield qualitatively the same results. There is no layer restriction (i.e., every connectivity following the feed forward paradigm is allowed and possible, see figure 1 for an example).

[1]The distinction between these two kinds of tasks is also discussed in [8]; it is shown that the fast learner should be preferred if it is necessary to cope with a class of problems, i.e., to be able to adapt to a number of different, but related problems, in particular within a short time. The modularity may be one of the main differences between the ANNs evolved in [8] for the two different tasks.

Training is performed by means of iRprop$^+$ [9], an improved version of the Rprop-Algorithm [12], which is a powerful, gradient-based learning algorithm for neural networks. The aim of training is the minimization of the mean squared error $E^{(\text{mse})}$ (i.e., the mean squared difference between the network's outputs $\hat{y}_j$ and the target values y_j). We compute the classification result of the network by mapping positive network outputs to "true" and negative ones to "false". The classification error $E^{(\text{class})}$ is given by the rate of incorrect classification outcomes.

3.3 Evolutionary Algorithm

The evolutionary algorithm is designed close to the evolutionary programming paradigm (EP) [5]. Prior to the first generation $\mu = 50$ individuals are initialized at random: The 60 connections are randomly spread over the whole network and the weights are initialized with values from the interval $[-0.2, 0.2]$. The only operators employed to generate the $\lambda = 50$ offspring per generation are random variations of the weights and random moves of single connections (i.e., to delete a connection and to insert it afterwards again at a random position). This means that the number of connections remains constant. This is an important point, as the number of connections has a strong impact on the modularity; in particular it has turned out that very modular ANNs ($\mathcal{M} = 1$) evolve in case of a slight selective pressure towards small ANNs. To avoid side-effects, we keep the size of the ANN constant. The parent population of the next generation is selected out of the parents and the offspring by means of the EP-tournament-selection [5], a rank-based selection method, with a tournament size of 5.

Depending on the goal of the optimization (accurate model or fast learner) the algorithms slightly differ. In the first case, we use Lamarckian inheritance; in each generation each individual is trained for 30 cycles and thereafter the modified weights are re-encoded into its genome. The fitness function includes only the error after learning:

$$f = E^{(\text{class})} + \alpha_2 \frac{E^{(\text{mse})}}{1 + E^{(\text{mse})}} \quad . \tag{6}$$

In case of the evolution of the fast learner, in each generation the weights of each individual are re-initialized randomly in the interval $[-0.2, 0.2]$ and learning takes place as long as there are still misclassified patterns left, but not for more than a maximum number of $T^{(\text{max})} = 1000$ cycles. In this case, the fitness is given

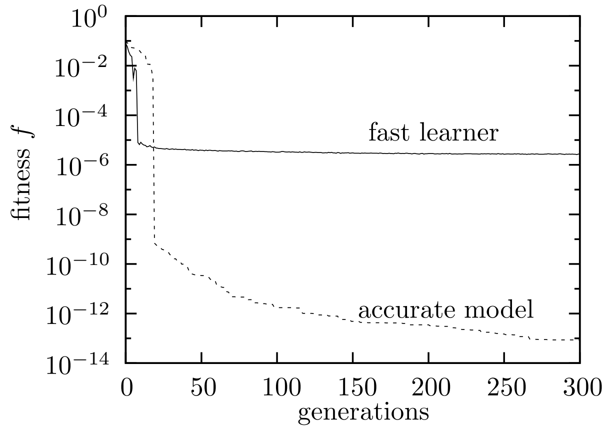

Figure 2: Trajectories of the fitness of the best individual, averaged over 20 independent runs per algorithm and per task. Due to the two different tasks, the absolute values of the fitness should not be compared.

by

$$f = E^{(\mathrm{class})} + \alpha_1 \frac{T^{(\mathrm{stop})}}{T^{(\mathrm{max})}} + \alpha_2 \frac{E^{(\mathrm{mse})}}{1 + E^{(\mathrm{mse})}} \quad . \qquad (7)$$

Herein, $T^{(\mathrm{stop})}$ denotes the cycle when learning has stopped. Also the parent's weights are re-initialized and, due to the influence of the random weight initialization to the individual's fitness, also their fitness is re-evaluated in each generation. Only mutations of the topology are performed, since no weights are included to the genome.

The choice of $\alpha_1 = 10^{-4}$ and $\alpha_2 = 10^{-8}$ effects that $E^{(\mathrm{class})}$ has the strongest and $E^{(\mathrm{mse})}$ the weakest influence to the fitness (i.e., the primary aim of the optimization is the correct classification).[2]

4 Discussion of Experimental Results

In this section, we present and discuss the results of our investigations.

Figure 2 depicts the evolution of the fitness of the best individual, averaged over 20 trials with different initializations. The absolute value of the fitness difference between the two tasks is not of major interest here, as it mainly reflects the different definitions (6) and (7) of the fitness function. However, it shows that it takes about 10 to 20 generations to evolve the first individuals solving the classification problem. Thereafter, the algorithms focus on the reduction of the learning time and the mean squared error, respectively.

4.1 Evolution of Modularity

The evolution of modularity during the optimization is of major interest in this article. In figure 3 (thick lines) we display the average modularity of the parent population. After initialization, the population's average modularity is close to $\mathcal{M}^{(\mathrm{paths})} \approx 0.5$, the expectation for randomly initialized ANNs, and $\mathcal{M}^{(\mathrm{neurons})} \approx 0.32$, respectively. Thereafter—except for some random fluctuations—all graphs show that the modularity increases, the more the error decreases. The significant increase of modularity supports the intuition that modular architectures are indeed advantageous for our task.

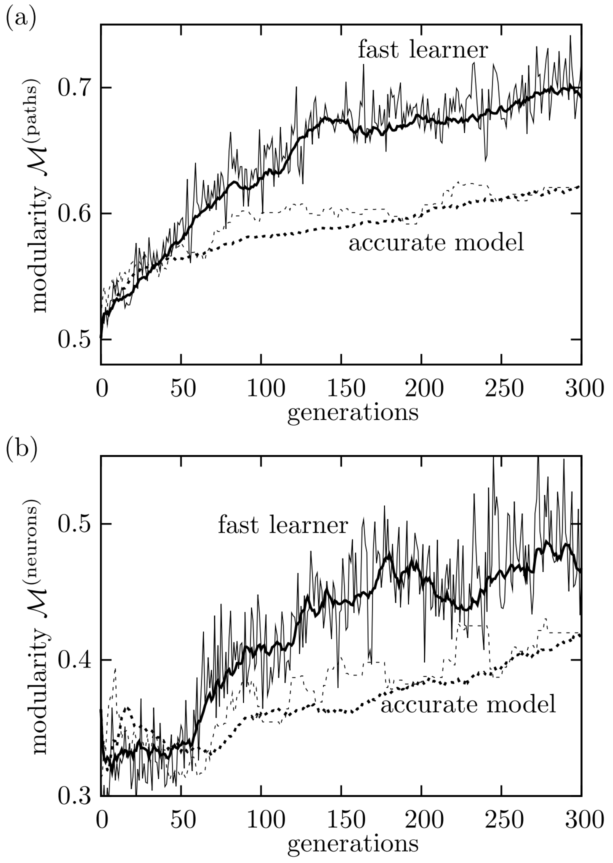

Figure 3: Evolution of the average modularity of the parent population (thick line) and of the modularity of the best individual (thin line), averaged over 20 independent runs per task. Figure (a) uses the path-count-measure, figure (b) the neuron-type-measure for the modularity.

[2]The magnitude of these parameters is arbitrary as long as they preserve the hierarchical order of the contributions to the fitness function (e.g., the correct classification is the primary goal and can not be leveled out by faster learning or a smaller $E^{(\mathrm{mse})}$).

However, it remains to explain why the topology does not evolve to a completely separated network of modularity $\mathcal{M} = 1$. Several reasons come to mind. One may assume that mutations will always produce an average offset from an individual with modularity of $\mathcal{M} = 1$. To acquire a rough idea of the magnitude of this offset consider the following table:

distance (mutations)	$\mathcal{M}^{(\text{paths})}$	$\mathcal{M}^{(\text{neurons})}$
0	1.00	1.00
1	0.96	0.93
3	0.88	0.82
5	0.82	0.75

It displays the average modularity—averaged over 100,000 trials—of a topology that is derived by 0, 1, 3, or 5 random mutations from a random topology with $\mathcal{M} = 1$. The table clearly demonstrates that an average modularity below 0.7 cannot be explained merely by a mutational offset from a completely separated topology. Further, figure 3 exhibits that the population's average modularity is not far from the best one's. The offset from modularity $\mathcal{M} = 1$ must have another origin.

Another explanation for the sub-extreme state is twofold, depending on the optimization task. In case of the fast learner, *learning*—in our case the very effective iRprop$^+$ algorithm—can partly counterbalance a non-optimal topology (e.g., a topology with high, but not maximum modularity). In particular, it is likely that an efficient learning algorithm will adjust the weight of an undesirable connection to a small absolute value.[3] In case of the accurate model optimization, the lack of modularity can be partly counterbalanced by the weights, which are also part of the evolutionary optimization. More illustrative: for the optimization it suffices when the weight of an undesirable 'cross-connection' is set close to zero *once* during evolution.

4.2 Modularity and Learning

Particularly interesting is the observation that the magnitude of the degree of modularity depends on the task. On average the optimization of the fast learner evolves topologies with a higher degree of modularity than the optimization of the accurate model (see figure 3 (a) and (b)). In case of the path-count-measure, the difference in modularity in the last generation is highly statistically significant (Wilcoxon rank sum test: $p \approx 0.84 \times 10^{-3}$), in the other case it is not significant (Wilcoxon rank sum test: $p \approx 0.13$).

Obviously, selection pressure towards modularity is not as predominant for the accurate model task, in which the trained weights are re-encoded. In this Lamarckian type of evolution, the acquired information is shared between the topological part and the weight part. This fact on its own explains a reduced selection pressure on the topological part. Further, the inheritance of the trained weights is advantageous for the offspring especially when the topology does not change. Thus, the algorithm tends to keep the topology rather stable and to take advantage of the continuous weight optimization during evolution. In contrast to this, the fast learner has to re-adjust the weights of the 'cross-connections' in every generation. Therefore, it seems to be more advantageous for the fast learner to remove these connections.

As already pointed out in section 4.1, this means that learning is not as efficient in counterbalancing topological imperfection as the evolution of suitable weights during the optimization. In other words: In our experiments, the demand for learning yields a stronger need for modular topologies.

4.3 On the Measures of the Degree of Modularity

Let us finally address the two measures itself. At first sight, figures 3(a) and 3(b) do not differ significantly. Both of the measures illustrate the increase of modularity. However, we find the second measure $\mathcal{M}^{(\text{neurons})}$ to be more fluctuating; the results of our experiments are less significant with respect to $\mathcal{M}^{(\text{neurons})}$. Still, we think that this measure is useful to quantify a more hierarchical type of modularity (e.g., when two disjoint sub-architectures referring to disjoint inputs share one output neuron, for example in case of processing and fusion of independent input signals.)

5 Conclusion and Outlook

In conclusion, we did not only confirm that the network performance depends on the network topology, but pinpointed the relationship between modular problems and modular topologies. We believe that it is important to investigate such relations in a quantitative manner in order to approach an understanding of

[3]Especially, the fact that we use a *batch* learning algorithm leads to the speculation that the problem's modularity is very quickly 'detected' by the learning algorithms since the gradient calculation averages over all possible cases and thereby all correlations between the subproblems are cancelled out. Further investigations should also consider an online learning algorithm in order to support or overrule this argument.

the development and functional importance of modularity. This is why we concentrated on a special character of network modularity, which can explicitly be quantified and related to a rather basic kind of problem modularity. As mentioned above, even on this restricted field further investigations are necessary.

From a broader perspective, all our efforts aim at the evolution of large systems with modular information processing. The present work is a small step towards understanding basic issues of modularity. In our opinion, further research should concentrate on at least two distinct subjects. First, recall that our measures of modularity neglect the kind of information processing within the neurons. Better measures of modularity have to be introduced which address not only the network topology but more directly the information flow. Such measures would guide the development of new types of neural networks being more modular *a priori* than today's common artificial neural networks.

A second subject of future research should concern the algorithmic point of the evolution of network topologies: If findings confirm the appropriateness of modular topologies for a class of problems, then evolutionary algorithms should be specifically designed to find modular topologies. New mutation operators or other types of encoding (e.g., recursive or grammar encodings [11, 7, 6, 14, 13, 18]) should be analyzed and (further) developed.

Acknowledgement

We would like to thank W. von Seelen for many beneficial discussions. Moreover we would like to thank the BMBF, grant LOKI, number 01 IB 001 C, and the DFG, grant Solesys, number SE251/41-1, for their financial support of our research.

References

[1] M. A. Arbib, editor. *The Handbook of Brain Theory and Neural Networks.* MIT Press, 1995.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 1995.

[3] J. L. Elman, E. A. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett. *Rethinking Innateness – A Connectionist Perspective on Development.* MIT Press, 1996.

[4] J. A. Fodor. *The Modularity of Mind: An Essay on Faculty Psychology.* The MIT Press, Cambridge, Massachusetts, 1983.

[5] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press, 1995.

[6] C. M. Friedrich and C. Moraga. An evolutionary method to find good building-blocks for architectures of artificial neural networks. In *Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems, IPMU '96*, pages 951–956, Granada, Spain, 1996.

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

[8] M. Hüsken, J. E. Gayko, and B. A. Sendhoff. Optimization for problem classes – Neural networks that learn to learn. In X. Yao and D. B. Fogel, editors, *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (ECNN 2000)*, pages 98–109. IEEE Press, 2000.

[9] C. Igel and M. Hüsken. Improving the Rprop learning algorithm. In H.-H. Bothe and R. Rojas, editors, *Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000)*, pages 115–121. ICSC Academic Press, 2000.

[10] M. I. Jordan and A. Jacobs. Modular and hierarchical learning systems. In Arbib [1].

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

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

[14] B. A. Sendhoff and M. Kreutz. Variable encoding of modular neural networks for time series prediction. In *Congress on Evolutionary Computation (CEC'99)*, volume 1, pages 259–266. IEEE Press, New York, 1999.

[15] H. A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106:467–482, 1962.

[16] N. Snoad and T. Bossomaier. MONSTER – the ghost in the connection machine: Modularity of neural systems in theoretical evolutionary research. In *Proceedings of the 1995 ACM/IEEE Supercomputing Conference*, 1995.

[17] P. Stagge and C. Igel. Structure optimization and isomorphisms. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, Natural Computing, pages 409–422. Springer-Verlag, 2001.

[18] M. Toussaint. Self-adaptive exploration in evolutionary search. Technical Report IRINI 2001-05, Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany, 2001.

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