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

Evolutionary Optimization for Problem Classes with Lamarckian Inheritance

Michael Hüsken, Bernhard Sendhoff

2000

Preprint:

This is an accepted article published in Seventh International Conference on Neural Information Processing – Proceedings. The final authenticated version is available online at: https://doi.org/[DOI not available]

Evolutionary Optimization for Problem Classes with Lamarckian Inheritance

Michael Hüsken

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

Bernhard Sendhoff

Future Technology Research (FTR)
HONDA R&D Europe GmbH
Carl-Legien-Straße 30
63073 Offenbach Main, Germany
bs@el-tec.de

Abstract

The combination of evolution and learning as two natural principles of optimization, which work on different time scales, has been shown to be very efficient for optimization tasks. We investigate how this combination should be organized to achieve networks that can fast and reliably switch between related problems from one and the same class of problems during operation time. We analyze different methods to evaluate the network's performance during optimization, namely a time averaged evaluation and a more direct averaging over different problems in each generation. In particular, we show that a specific kind of Lamarckian inheritance can be beneficial for the evolutionary optimization of neural networks even for dynamic environments like problem classes.

1 Introduction

The ability to learn and subsequently to generalize is one of the most distinguished features of models of neural information processing like artificial neural networks (NNs) and in particular multilayer perceptrons (MLPs). The question how the learning capacity can be increased and generalized has recently received much attention in particular in the machine learning community [1]. Most learning algorithms are based on statistics, i. e. they rely on a large amount of data. However, humans can learn tasks and coherences from very few (or even one) examples.

Of course, if we strive to augment statistical learning with other methods, we have to be aware that the information for the decision of the neural systems, e. g. for a control task, has to be supplied from additional sources. Examples are information from related tasks in multitask learning [2] or from information available in a different representation as in fuzzy systems [3].

A second approach aims at an appropriate predisposition of the NN to achieve learning of various different albeit related tasks. In this way, we can achieve sequential rapid learning of tasks on an appropriate initial structure based on few additional data. At the same time *unwanted* interference of newly acquired and already represented information in the neural system has to be avoided.

The second approach belongs to the field of evolution and learning. In particular in this paper, we will deal with the evolutionary aspect of the "learning to learn" paradigm. In the notation of the classification introduced by Thrun and Pratt [1, chapter 1], this would belong to the area of representations and functional decompositions. However, in the evolutionary approach, introduced in [4], this functional decomposition is self-organized during evolution. Thus, common aspects are extracted during evolution and genetically fixed in a specialized topology and weight initialization of the neural system due to a selective advantage. Thus, genetically provided information forms the basis for the ability of the NN to learn particular problems from a *class of related problems*. At the same time, the information acquired during lifetime influences the evolutionary process, either direct – Lamarckian evolution – or indirect through the Baldwin effect. It has been argued before that Lamarckian inheritance does not result in robust neural networks that can cope with dynamic environments. We will show that this is not the case when the Lamarckian inheritance of "lifetime" information is organized in such a way that only *average* knowledge is transfered.

Developing NNs that *learn* during evolution to *learn* during lifetime is a particular constraint on the evolutionary optimization of NNs. Apart from the biological plausibility, there is also a strong application need for NNs that can cope with related problems during their operation time in a fast and reliable manner based on only a few additional data. In particular in safety sensitive application domains like the automotive industry, the adaptation of a NN controller to new albeit related situations must occur without noticeable time consumption. An example problem, the system state diagnosis with a neural network, is described in [4].

In the next section, we will outline the different approaches that can be utilized to develop NNs

that can generalize between problems belonging to one class. In Section 3 we will discuss experimental results and in the last section we will draw our conclusions.

2 Coping with related problems

As we pointed out in the introduction, the design of NNs that can cope with related problems, thus that have "learnt to learn" during the evolutionary optimization, was largely motivated by the need for particular applications [4]. Of course, the benefit of using the EANN paradigm in this framework should be the reduction of computation time during adaptation.

Additionally, the problem class should be open during the NN's operation, i. e. problems that have not been encountered during the evolutionary optimization, however that belong to the same class, should also be covered by the NN. Similar to the usual generalization of NNs between different data sets from one and the same problem, we term this capacity *second order generalization*. Therefore, second order generalization is necessary to ensure the ability to adapt to all problems of the class, even to those that have not been seen during the evolutionary optimization.

Second order generalization can be achieved in a similar way as first order generalization in NN 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.

2.1 Development of second order generalization during evolution

There are two different strategies, which can be used to achieve second order generalization during evolutionary optimization.

Ensemble-Method: Each individual NN is tested in *each* generation on a number of problems from the same class to estimate the *second order adaptability*, i. e. the ability to learn all problems of the problem class. Therefore, the error measure (the inverse of the fitness) is a combination of the errors for each single problem and the selection pressure is *directly* towards coping with a problem class; this is depicted in Figure 1 (b).

Generation-Method: Another approach is to change the task in different generations, i. e. to select the best individual in generation t for problem i_t and in generation $t+1$ for a different problem i_{t+1}, see e. g. [5, 6]. Therefore, in the Generation-Method, see Figure 1 (c), the selection pressure is *indirectly* towards coping with a problem class. However, we will see in later sections that it can also be very effective and in general it will require fewer additional NN evaluations.

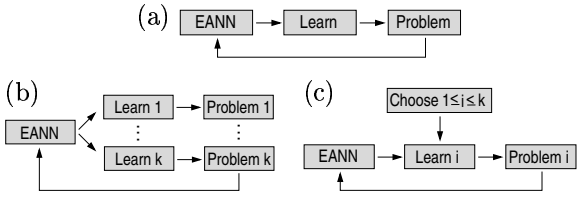

Figure 1: Different designs of algorithms for the evolutionary optimization of NN (EANN). Algorithm (a) is mostly used for the design of specialized NNs. The Ensemble-Method (b) and Generation-Method (c) present different strategies for the development of second order adaptable NNs, which can cope with a class of problems.

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 a priori completely unknown problems during lifetime.

2.2 Lamarckian inheritance for problem class optimization

In the case of optimization for one particular problem, the combination of evolution and learning, see Figure 1 (a), has turned out to be very fruitful. In particular, the Lamarckian scheme of inheritance, i. e. re-encoding the modified weights after the period of learning, has been beneficial for the optimization of NNs (e. g., SEV [7], ENZO [8], and EPNet [9]). Therefore, it is desirable to exploit these benefits from the Lamarckian inheritance also for the evolutionary optimization of NNs for a class of problems. At first sight, this seems to contradict the demand for an increased flexibility of the NN's structure in a changing environment, which is represented by the open problem class. However, the *averaged Lamarckian inheritance*, which was proposed in [4], is a means to combine the need for flexibility with the advantage of a head start for learning due to the more refined initial weight setting.

First, we focus on the Ensemble-Method. To take advantage of the knowledge gained during lifetime, 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 denoted by i. If one particular weight turns out to have similar values after adaptation independent of i, this value seems to be an appropriate 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 re-encode the arithmetic mean of the weights resulting from learning the different problems. Let $w_{m,n}^{(i)}$ be the weight of the connection from neuron m to neuron n after learning the problem i. The weight for this connection to be re-encoded in generation t is given by

$$w_{m,n}^{t} = \frac{1}{k} \sum_{i=1}^{k} w_{m,n}^{(i)} \quad . \tag{1}$$

Here, k denotes the number of problems taken into account during the evolutionary optimization.

The averaged Lamarckian inheritance for the Generation-Method is very similar. The main difference of this method compared to the Ensemble-Method is that the second order adaptability is not evaluated in each generation, but implicitly averaged over succeeding generations. Only individuals that have the ability to learn are able to create more offsprings. Therefore, it is natural to carry over the averaged Lamarckian inheritance by introducing a *time-averaged Lamarckian inheritance*. In this method the average (1) is calculated over succeeding generations:

$$w_{m,n}^{t} = (1-c) \cdot w_{m,n}^{t-1} + c \cdot w_{m,n}^{(i_t)} \tag{2}$$

The variable $w_{m,n}^{(i_t)}$ gives the value of the weight $w_{m,n}$ after learning problem i_t in generation t. The choice of the parameter c is related to the lifetime of the influence of the weights in the past generations. In the following, we present an estimation of the value of c.

Since we are interested how strong the weight $w_{m,n}^{t}$ is influenced by its value $w_{m,n}^{t-\Delta t}$ from Δt generations ago, (2) is rewritten as

$$w_{m,n}^{t} = (1-c)^{\Delta t} \cdot w_{m,n}^{t-\Delta t} + c \cdot \sum_{\Delta\tau=0}^{\Delta t-1} (1-c)^{\Delta\tau} \cdot w_{m,n}^{(i_{t-\Delta t})} \quad . \tag{3}$$

If we estimate the weights by an average $\bar{w}_{m,n}$, we get

$$\begin{aligned} w_{m,n}^{t} &\approx \bar{w}_{m,n} \left[(1-c)^{\Delta t} + c \sum_{\Delta\tau=0}^{\Delta t-1} (1-c)^{\Delta\tau} \right] \\ &= \bar{w}_{m,n} \left[(1-c)^{\Delta t} + (1-(1-c)^{\Delta t}) \right] \quad . \end{aligned} \tag{4}$$

The relative influence ξ of the old information can now be estimated by the fraction between the left (old) and the right (new) term in (4):

$$\xi \approx \frac{(1-c)^{\Delta t}}{1-(1-c)^{\Delta t}} \tag{5}$$

The lifetime of the information is given by the time needed to decrease ξ below $1/e$. Therefore, c is given by

$$c \approx 1 - \left(\frac{1}{e+1} \right)^{1/\Delta t} \quad . \tag{6}$$

We suggest to choose the lifetime Δt equal to the number of different environments (number of different problems) k during the evolutionary process. Therefore, it is equal to the average time for one particular environment to appear again.

2.3 Implementation details

The NN's weights and topology are encoded with a direct encoding scheme. The mutations of the individuals are close to the suggestions in [8] (i. e., insertion/deletion of connections/hidden neurons, no crossover). Learning is performed using iRprop$^+$, an improved version of the Rprop-Learning algorithm [10], for 300 learning cycles. To avoid over-fitting, learning is combined with cross-validation. In case of the Ensemble-Method the EP-Tournament-Selection [11] is used and due to the evolution in changing environment in case of the Generation-Method, we employ the non-elitistic (μ, λ)-selection.

3 Experimental results

In our experiments we mainly focus on the benefit of the Lamarckian inheritance in the optimization for problem classes. We utilize the artificial regression problem class, introduced in [4]. This problem class is given by the superposition

$$f_a(\boldsymbol{x}) = a\tilde{f}_1(\boldsymbol{x}) + (1-a)\tilde{f}_2(\boldsymbol{x}) \ , \ \boldsymbol{x} \in [0,1]^8 \tag{7}$$

of the two functions

$$\tilde{f}_1(\boldsymbol{x}) = \frac{1}{2+\sqrt{2}} \left[x_1 + \sqrt{x_2^2 + x_3^2} + \frac{\sin\left(\pi \frac{x_4 x_5}{2}\right)}{x_6 + 1} \right] \tag{8}$$

and

$$\tilde{f}_2(\boldsymbol{x}) = \frac{2}{4+\sqrt{2}} \left[\sqrt{\frac{1}{2} x_2 x_3} + \cos\left(\pi \frac{\sqrt{x_4^2 + x_5^2}}{4} \right) + (x_6 - x_7)^2 \right] \quad . \tag{9}$$

For each value of a one particular regression problem is given, which is represented by four different data sets.

During the evolutionary optimization $k = 5$ different problems of the problem class are given; later the second order generalization of the NNs is tested on 11 different problems.

3.1 Ensemble-Method

First, we focus on the averaged Lamarckian inheritance. In our first experiment, the influence of the scheme of inheritance for the progress of the optimization is investigated. Therefore, we combine the Ensemble-Method with different schemes of inheritance and different codings:

Ense$\mathbf{M}_1$: Only the NN's topology is encoded and the initial weights are chosen at random in every generation.

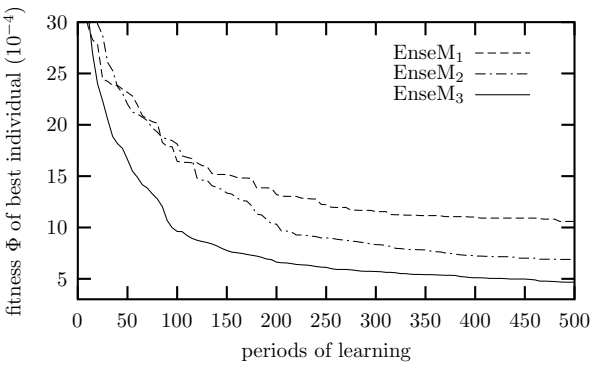

Figure 2: Temporal development of the individual's fitness for different schemes of inheritance in the Ensemble-Method.

EnseM$_2$: Both, the topology and the initial weights are encoded. The inheritance follows the Darwinian scheme, i. e. the weights are not re-encoded and only modified by means of mutations.

EnseM$_3$: As in EnseM$_2$, the topology and the initial weights are encoded. Furthermore, the averaged Lamarckian inheritance is used.

We should mention, that all of these algorithms operate on the same search space and are in principle able to find the same solutions. However, they differ in the way in which they exploit information from the individual's lifetime.

The results, averaged over 10 independent runs, are given in Figure 2. The comparison of EnseM$_1$ and EnseM$_2$ indicates the importance of optimizing the weights together with the topology. However, the temporal development of the fitness in case of EnseM$_3$ shows the clear benefit of the more systematic approach of the averaged Lamarckian inheritance for the optimization of the weights.

The averaged Lamarckian inheritance does not only improve the optimization of the initial weights, but also feeds back beneficially to the optimization of the NN's topology. This is shown in a second experiment. We apply the optimized NNs to the task of learning the different problems of the class. The mean squared error after 300 cycles[1] of learning the particular problems for different values of a are given in Figure 3 (a). As it was to be expected from Figure 2, the NNs resulting from EnseM$_3$ perform best on the different problems. It should be noted, that the performance of the NN resulting from EnseM$_3$ is comparable to an ensemble of NNs that are optimized for different particular problems of the problem class. The errors are normalized such that an error of one is equal to the performance of a specialist for this problem.

[1]This number of cycles has turned out to be sufficiently long to learn different problems and satisfies the demand of a fast adaptation. Learning is combined with cross validation to avoid over-fitting.

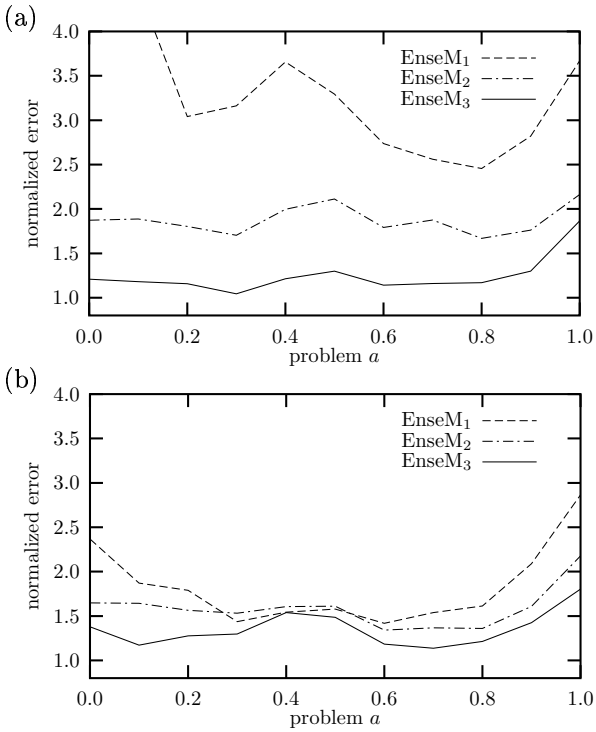

Figure 3: Mean squared error after learning the different problems of the problem class for a fixed time. (a) Learning starts from the topology and initial weights resulting from the Ensemble-Method and in (b) learning starts from the same topology with randomly initialized weights. The error is normalized to the error of a NN of approximately the same size that is specialized to this particular problem a.

The feedback of the averaged Lamarckian inheritance to the optimization of the topology of the NN can be observed by re-initializing the weights and lengthening the period of learning to 5000 cycles. Due to this re-initialization, only one part of the optimization result – the topology – is kept and evaluated. Since the time for learning is clearly longer, only the *ability* to represent the different problems of the class is investigated. Because of the random initialization of the weights, learning often got stuck in local minima. Therefore, only the best 10 runs out of 50 initializations per topology are taken into account for the results presented in Figure 3 (b).

EnseM$_1$ and EnseM$_2$ perform better than before due to the enlarged period of learning. However, the more interesting point is, that the ranking of the results of the three algorithms remains unchanged: EnseM$_3$ still performs best, although all NNs are re-initialized and therefore cannot take advantage of the kind of weight optimization during the evolution. This emphasizes the feedback between the optimization of initial weights and of the topology. Since the optimization of the weight initialization is more successful with the averaged Lamarckian

algorithm	(μ, λ)	$\bar{\phi}$	$\phi_{\min}$	σ_ϕ	$\bar{\Phi}$	$\Phi_{\min}$	σ_Φ
GeneM_2	(2,15)	7.4	2.2	5.2	9.2	5.4	4.9
GeneM_2	(5,15)	6.1	2.0	3.2	7.2	5.1	1.7
GeneM_2	(8,15)	7.4	2.7	3.9	8.8	6.0	2.5
GeneM_3	(5,15)	4.8	1.4	2.9	5.7	4.4	1.3

Table 1: Variations of the individual's fitness ϕ as well as the lack of adaptability Φ starting from the 500th Generation in the Generation-Method. All values have to be multiplied by 10^{-4}.

inheritance, the selection of the best NN topology also becomes more reliable. In this way, the topology can be more attuned to the weight setting.

3.2 Generation-Method

In this section, we present results, how the time averaged Lamarckian inheritance improves the Generation-Method. In our experiments we compare results from the following two kinds of Generation-Methods that only differ in the scheme of inheritance. In both cases, the topology and the initial weights are encoded in the genome.

GeneM_2: The trained weights are not re-encoded and the initial weights are only modified by means of random mutations.

GeneM_3: The time averaged Lamarckian inheritance is used. Following equation (6) and setting the lifetime equal to the number of different environments ($\Delta t = 5$), one gets $c = 0.23$.

Due to the non-elitistic selection scheme, after about the 300th generation the evolutionary process stagnates and only random fluctuations are observed. The mean of these fluctuations $\bar{\phi}$, the minimum fitness $\phi_{\min}$ as well as the standard deviation σ_ϕ, averaged over 10 independent runs, are given in Table 1. The dependency of these values on the selection pressure can be understood: With increasing selection pressure it becomes more important for being selected to solve the current problem with a low error. Individuals with the ability to solve different problems moderately well will not remain in the population. Therefore, no averaging over time takes place and $\bar{\phi}$ and σ_ϕ increase. In case of a very low selection pressure, averaging over time takes place, but the pressure to optimize the ability to learn is low. Therefore, σ_ϕ decreases compared to the case of a high selection pressure, but $\bar{\phi}$ remains high. For a moderate selection pressure the best results are observed, since both an averaging of the adaptability over time and a pressure to reduce the adaptability occur.

In [4] it was shown, that in case of the generation method low fitness is not equivalent to good adaptability to all problems of the problem class. Better results are achieved by measuring the error Φ after learning a number of problems of the problem class during the evolutionary run. To reduce the amount of computation, Φ is only calculated for the parent population in every fifth generation. In analogy to ϕ, the decrease of Φ stops after about 300 generations. The explanation of the dependency of the remaining variations on the selection pressure (see Table 1) is identical to the one for ϕ.

In our context, the interesting point is the decrease of the mean fitness $\bar{\phi}$ and the mean lack of adaptability $\bar{\Phi}$ for GeneM_3 compared to GeneM_2, which is caused only by the time averaged Lamarckian inheritance. Moreover, the decrease of the standard deviation of the fluctuations σ indicates a more stable evolutionary process.

At first sight, these results are contradictory to results in [6, 12]. There it is stated, that Lamarckian inheritance in a changing environment destabilizes the evolutionary process. However, the experimental setup in their experiments was quite different from our environment. On the one hand, we focus on the optimization of both, the topology and the weights. This softens the specialization of the individuals, as the topology changes on a larger time scale compared to the changes of the environment and the adaptation of the weights during lifetime. On the other hand, the changes in our environment are faster, but more continuously compared to [6, 12]. Therefore, on the one hand individuals have not the possibility to specialize to one problem in succeeding generations and on the other hand knowledge from the previously solved problem becomes more supportive. Finally, our evolutionary approaches have to cope with a much lower number of fitness evaluations, so that a random mutative search in the weight space would be insufficient.

Of course, from the practical point of view the performance of the best NN is of more interest than the stability of the evolutionary process. In Table 1 it can be found, that both the best fitness ϕ and the lowest average error after learning a number of problems Φ is decreased due to the time-averaged Lamarckian inheritance. As pointed out previously, the reduction of Φ is of particular interest. In Figure 4 (a) the NNs with the lowest value of Φ are investigated. The ranking of the NNs are the same as indicated by the values of $\bar{\Phi}$ from Table 1. In particular, the improvement of the result due to the re-encoding of the weights becomes significant. In Figure 4 (b) it can be seen that the Generation-Method with time-averaged Lamarckian inheritance outperforms all other results. If we keep in mind, that an error of one corresponds to the error of a specialized NN of approximately the

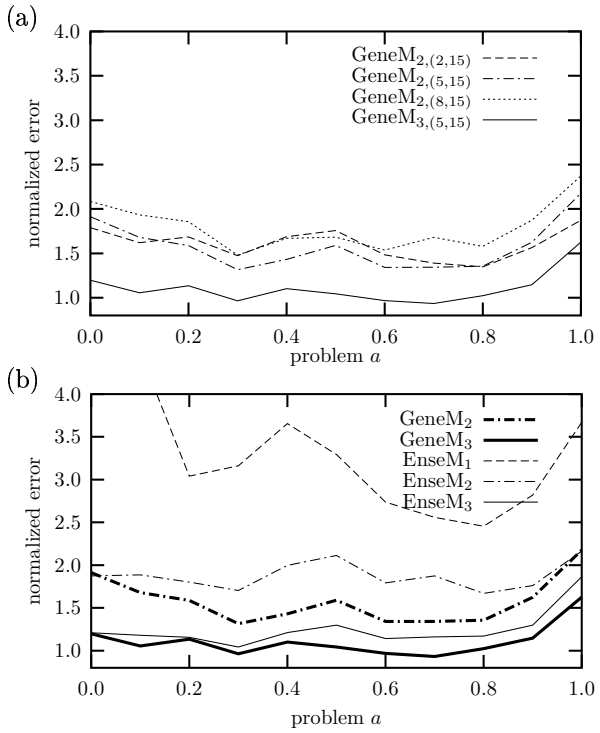

Figure 4: Mean squared error after learning the different problems of the problem class for a fixed time. (a) Comparison of the NNs resulting from the Generation-Method without (GeneM_2) and with (GeneM_3) time-averaged Lamarckian inheritance. In diagram (b) the algorithms with the $(5, 15)$-selection scheme are compared with the results of the Ensemble-Method.

same size, this result is very close to the optimum attainable by an ensemble of specialists.

4 Conclusion

This paper deals with the evolutionary optimization of NNs with respect to a class of related problems. Therefore, during evolution a predisposition of the structure and the weight initialization of the NN is found so that it can learn related problems fast and reliably based on few additional data and little learning time. Two hierarchical methods are proposed to achieve this predisposition, which both aim at the optimal combination, coupling and information transfer between evolution and learning. We have empirically shown that Lamarckian inheritance with a twist, i. e. to only code an *average* knowledge about the environment back to the genetic representation, is superior to a purely Darwinian feedback *even* in dynamic environments. Both the Ensemble-Method and the Generation-Method, which were analyzed as two different approaches to evaluate the generalization of networks between problems that belong to the same class, show better performance for the averaged Lamarckian inheritance. It is interesting to note that the indirect evaluation of the networks using the Generation-Method is slightly better than the more direct Ensemble-Method. Thus, to conclude, averaging over time and problems with Lamarckian inheritance can produce the most robust neural networks in dynamic environments for the problems that were examined in this work.

Acknowledgment
We would like to thank J. Gayko and W. von Seelen for fruitful discussions. Part of this work is supported by the BMBF grant LEONET (01 IB 802 C4). Furthermore, we would like to thank the Research Centre Jülich for the processing time on the Cray T3E system.

References

[1] S. Thrun and L. Pratt, editors. *Learning to Learn.* Kluwer Academic Publishers, 1998.

[2] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[3] Y. Jin and B. Sendhoff. Knowledge incorporation into neural networks from fuzzy rules. *Neural Processing Letters*, 10(3):231–242, 1999.

[4] M. Hüsken, J. E. Gayko, and B. Sendhoff. Optimization for problem classes – Neural networks that learn to learn. In *Proceedings of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks.* IEEE Press, 2000.

[5] S. Nolfi, O. Miglino, and D. Parisi. Phenotypic plasticity in evolving neural networks. In *Proceedings of the First International Conference From Perception to Action*, pages 146–157. IEEE Computer Society Press, 1994.

[6] T. Sasaki and M. Tokoro. Adaptation under changing environments with various rates of inheritance of acquired characters: Comparison between Darwinian and Lamarckian evolution. In *Simulated Evolution and Learning.* Springer, 1998.

[7] R. Lohmann. Structure evolution and incomplete induction. *Biological Cybernetics*, 69:319–326, 1993.

[8] H. Braun and P. Zagorski. ENZO-M – a hybrid approach for optimizing neural networks by evolution and learning. In Y. Davidor, H.-P. Schwefel, and R. Maenner, editors, *Proceedings of the third International Conference on Parallel Problem Solving from Nature*, pages 440–451, 1994.

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

[10] C. Igel and M. Hüsken. Improving the rprop learning algorithm. In *Proceedings of the Second International ICSC Symposium on Neural Computation*, pages 115–121. ICSC Academic Press, 2000.

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

[12] T. Sasaki and M. Tokoro. Adaptation toward changing environments: Why Darwinian in nature? In *Fourth European Conference on Artificial Life*, pages 145–153. MIT Press, 1997.