ELSEVIER
Neurocomputing ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Operator adaptation in evolutionary computation and its application to structure optimization of neural networks

Christian Igel[a,*], Martin Kreutz[b]

[a] *Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany*
[b] *ZN Vision Technologies AG, 44781 Bochum, Germany*

Received 22 October 2001; accepted 8 March 2002

Abstract

The problem of finding a suitable neural network topology for a given task is often solved by evolutionary computation. In this paper, we show empirically that online adaptation of the search strategy can increase the performance of evolutionary structure optimization. After a brief overview of strategy adaptation in evolutionary computation, we present a general method for adjusting the probabilities of applying variation operators. In example problems the adaptation method leads to faster optimization and better solutions when used in structure optimization of neural networks. We observe that during the evolutionary process the operator probabilities change in an intuitive way depending on the task. © 2002 Published by Elsevier Science B.V.

Keywords: Evolutionary algorithms; Feed-forward neural networks; Operator probabilities; Strategy adaptation; Structure optimization

1. Introduction

The performance of a neural network (NN) depends crucially on the underlying network topology. Finding a suitable topology for a given task constitutes a difficult structure optimization problem, which has been tackled successfully with evolutionary algorithms (EAs) [59,1]. Various EAs for structure optimization exist, but currently there is no constructive way of adjusting the parameters of these algorithms for a given problem, except some general design heuristics [20,54]. However, the ability of

[*] Corresponding author. Tel.: +49-234-32-25558; fax: +49-234-32-14209.
E-mail addresses: christian.igel@neuroinformatik.ruhr-uni-bochum.de (C. Igel), martin.kreutz@zn-ag.de (M. Kreutz).

0925-2312/02/$ - see front matter © 2002 Published by Elsevier Science B.V.
PII: S0925-2312(02)00628-8

an evolutionary algorithm to adapt its search strategy during the optimization process is an important concept in evolutionary computation, see the overviews [3,51,19]. In this article, a derandomized algorithm that adapts the probabilities of applying variation operators is proposed and evaluated in the context of structure optimization of NNs.

In Section 2, we give a brief survey of strategy adaptation in EAs and discuss strategy adaptation within the framework of global random search. In Section 3, a method for adjusting operator probabilities is described. In Section 4, we compare algorithms with adaptive and fixed operator probabilities applied to structure optimization of NNs.

2. Strategy adaptation

Throughout this article, we denote the setting of an EA (i.e., the choice of the variation operators and their parameters, the population size, …) as its search strategy. The variables that parameterize the space of search strategies are called strategy parameters. Online adaptation of these parameters (i.e., during optimization) is important because the best setting of an EA is not known a priori for a given task and the optimal search strategy is usually not constant during the evolutionary process. The latter point is of particular importance in the case of non-static fitness landscapes, as considered in [33].

Evolutionary algorithms can be regarded as a class of global random search algorithms. Let $\mathscr{G}$ denote the search (genotype) space and $\Phi : \mathscr{G} \to \mathbb{R}$ a quality (fitness) function. The general global random search scheme can be described as follows:

1. Choose a joint probability distribution $P^{(t)}_{\mathscr{G}^\lambda}$ on $\mathscr{G}^\lambda$. Set $t := 1$.
2. Obtain λ points $\boldsymbol{g}^{(t)}_1, \ldots, \boldsymbol{g}^{(t)}_\lambda$ by sampling from the distribution $P^{(t)}_{\mathscr{G}^\lambda}$. Evaluate Φ (perhaps with random noise) at these points.
3. According to a fixed (algorithm dependent) rule construct a new probability distribution $P^{(t+1)}_{\mathscr{G}^\lambda}$ on $\mathscr{G}^\lambda$.
4. Check for some appropriate stopping condition; if the algorithm has not terminated, substitute $t := t + 1$ and return to step 2.

In an EA, the search distributions $P^{(t)}_{\mathscr{G}^\lambda}$ are defined by the population[1] and the search operators.[2] Sampling the distribution corresponds to generating offspring. Constructing a new probability distribution in step 3 corresponds to selection and the update of the search strategy. From this point of view, altering the population and altering the search operators are of equal importance.

[1] There is a growing interest in (evolutionary) algorithms that adapt search distributions not based on the concept of populations [11,35,38].

[2] In some EAs, the distribution $P^{(t)}_{\mathscr{G}^\lambda}$ can be factorized as

$$P^{(t)}_{\mathscr{G}^\lambda}(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_\lambda) = P^{(t)}_{\mathscr{G}}(\boldsymbol{g}_1; \boldsymbol{\theta}^{(t)}) \cdots P^{(t)}_{\mathscr{G}}(\boldsymbol{g}_\lambda; \boldsymbol{\theta}^{(t)}). \tag{1}$$

This means, the offspring are created independently of each other based on the same distribution. In this case, the above scheme corresponds to global random search as defined in [60]. In general $P^{(t)}_{\mathscr{G}^\lambda}$ cannot be factorized as in (1), e.g., when crossover producing multiple offspring is employed or each parent generates exactly one offspring as in Section 4.1.

Strategy adaptation methods can be categorized roughly according to three major questions [51]:

What is being adapted? In most cases, the parameters of operators are adjusted, e.g., the mutation distribution (in evolution strategies [41,49,42,50,37,25], in evolutionary programming [23], or in genetic algorithms by adapting the mutation probabilities [8]), recombination [48,53] including *linkage learning* [26], or acceptance levels in the employed selection operator [46]. Other parameters that have been adapted include the probabilities of applying operators [17,18,34,24,16,56,30,33,28,36], the population size [52,27], the lifetime of individuals [7], and parts of the encoding [47].

What is the scope of adaptation? Strategy parameters can refer to different levels; they can affect the whole population, single individuals, or components of single individuals. Adaptation at the level of the individual is reasonable if different regions in the search space require different search strategies and there is enough diversity in the population for different regions to be explored at the same time. Examples of the adaptation at the component level are the adjustment of individual step-sizes in evolution strategies and the control of the mutation probabilities of the components of finite state machines in evolutionary programming [5]. The level depends strongly on what is being adapted—for instance the population size should be adapted at the population level. However, the covariance matrix adaptation proposed in [25] adapts at the population level the step sizes, which refer to components of individuals.

How are the changes made? This question can be further subdivided into two questions: What is the evidence upon which the adaptation is based and how is the change of the search strategy carried out? The strategy parameters can be altered in a deterministic or stochastic way. The adaptation process can be determined by an external fixed schedule or by a heuristic based on information gathered during the evolutionary process, see Section 3. An important concept is self-adaptation [9], which is used in all main paradigms of evolutionary computation [48,23,8,42,50,22,4]: Some strategy parameters are part of each individual and are subject to the same selection process as the individuals. They can be altered either stochastically by means of mutation and recombination or in a deterministic manner.

What little theory exists on strategy adaptation in EAs is mostly in the field of evolution strategies, e.g., there are recent investigations of the influence of self-adaptation on the population mean and variance [12] and convergence [45]. It has been shown empirically that the combination of different adaptation strategies can be beneficial [10]. Recently, the theoretical link between self-adaptation and non-injective genotype–phenotype mappings has been established [55].

3. An algorithm for the adaptation of operator probabilities

If different operators are employed in an EA, the probabilities of their application (also called operator fitnesses [18] or operator probabilities) can be regarded as strategy parameters, which are suitable for online adaptation. The first step is to judge the performance of the operators. In most cases, the performance measure is somehow related to (recent) fitness improvements produced by the operator [17,18,34,24,30,33,28,36].

However, other heuristics are reasonable, e.g., in [21] it is suggested to increase the mutation rate in cases of a loss of diversity in the population.

An adaptation scheme for operator probabilities based on a performance measure should fulfill the following basic requirements: An operator that has performed better than another for a certain number of generations should have a higher operator fitness; equal performance of two operators should lead to equal operator fitness values. There should be a minimum probability, say $p_{\min}$, for each operator to be applied, so that no operator, which might be useful at a later stage of the optimization process, may become extinct. Further, random fluctuations of the probabilities should be damped. The algorithm presented in the remainder of this section takes these requirements into account.

The goal is to maximize a fitness function Φ. Let Ω be the set of variation operators and let $p_o^{(t)}$ be the probability that $o \in \Omega$ is chosen at generation t. We consider only asexual operators (see [33] for a brief discussion of sexual or panmictic operators in this context) and suppose that each offspring is generated in the following way: First, a parent is reproduced. Then, the number of mutation operators to be applied consecutively to the reproduced individual is determined. This number can be constant or given by a random variable. Then the operators to be applied are selected independently from Ω at random, where the operator o has the probability $p_o^{(t)}$.

Let $\mathcal{O}_o^{(t)}$ contain all offspring produced at generation t by an application of the operator o. If an offspring is produced by applying more than one operator, then there is a "credit assignment" problem. We suggest treating this case as if the offspring has been generated several times, once by each of the operators involved. For example, if individual $\boldsymbol{g}$ has been generated by consecutive application of the operators o_i and o_j, then $\boldsymbol{g}$ is added to $\mathcal{O}_{o_i}^{(t)}$ and $\mathcal{O}_{o_j}^{(t)}$.

A quality measure for operators can be based on an elemental real-valued function q that measures the value of a single modification. One possible choice for such a measure, which depends only on the fitness values of the generated offspring $\boldsymbol{g}$ and of its parent, has been termed *benefit* [56] and is given by

$$q_B(\boldsymbol{g}) := \max\{\Phi(\boldsymbol{g}) - \Phi(\text{parent}(\boldsymbol{g})), 0\}. \tag{2}$$

The function parent : $\mathcal{G} \to \mathcal{G}$ returns simply the parent for a given offspring. This definition, which is used for operator adaptation in [56,28], can be regarded as a combination of the probability of improvement and the expected improvement [42,50]. It is important to consider not only the probability of beneficial steps: An operator that rarely generates large improvements should be rated similarly as an operator that produces small improvements frequently.

Another performance measure can be defined as

$$q_{B^*}(\boldsymbol{g}) := \max\{\Phi(\boldsymbol{g}) - \Phi(\boldsymbol{g}_{\text{best}}), 0\}, \tag{3}$$

where $\boldsymbol{g}_{\text{best}}$ is the best individual in the current parent population. This measure corresponds to Davis' *local delta* or *credit* [17,18] and the *absolute benefit* [33]. It links the benefit to evolvability [2] in the sense that the ability to produce better offspring than the best in the population is rewarded. The absolute benefit has the following major drawback: If the problem is difficult for the EA and offspring that are better than the best are rare, then the absolute benefit is determined by only a few events and the empirical evidence for the adaptation may become unreliable.

As an addition to the rhs of Eqs. (2) and (3), the computational effort of each operator in question has to be considered [33,29], i.e., the quality must be normalized by the computational costs incurred to achieve it (which may be measured in fitness evaluations).

In our operator adaptation algorithm, the operator probabilities are updated every τ generations. We call this period an adaptation cycle. The quality of an operator o during τ generations measured after generation t is given by

$$q_o^{(t,\tau)} := \frac{\sum_{i=0}^{\tau-1} \sum_{\boldsymbol{g}\in\mathcal{O}_o^{(t-i)}} q(\boldsymbol{g})}{\sum_{i=0}^{\tau-1} |\mathcal{O}_o^{(t-i)}|}. \tag{4}$$

This means, $q_o^{(t,\tau)}$ measures the average performance achieved by the operator o over an adaptation cycle. It is convenient to define the quality of all employed search operators during a time interval τ:

$$q_{\text{all}}^{(t,\tau)} := \sum_{o'\in\Omega} q_{o'}^{(t,\tau)}. \tag{5}$$

Now we can formulate our algorithm for the adaptation of the operator probabilities. For $t > 0$ the probabilities $p_o^{(t+1)}$ can be calculated as follows:

```
If $t \bmod \tau = 0$ then
    for each $o \in \Omega$ do
        $\tilde{p}_o^{(t+1)} := \begin{cases} \zeta \cdot q_o^{(t,\tau)}/q_{\text{all}}^{(t,\tau)} + (1-\zeta)\cdot \tilde{p}_o^{(t)} & \text{if } q_{\text{all}}^{(t,\tau)} > 0 \\ \zeta/|\Omega| + (1-\zeta)\cdot \tilde{p}_o^{(t)} & \text{otherwise} \end{cases}$
    for each $o \in \Omega$ do
        $p_o^{(t+1)} := p_{\min} + (1 - |\Omega| \cdot p_{\min}) \frac{\tilde{p}_o^{(t+1)}}{\sum_{o'\in\Omega} \tilde{p}_{o'}^{(t+1)}}$
else
        $p_o^{(t+1)} := p_o^{(t)}$
```

Here, $\tilde{p}_o^{(t+1)}$ is some kind of weighted average of the quality of the operator o, where the influence of previous adaptation cycles decreases exponentially. The rate of this decay is controlled by $\zeta \in (0,1]$, where $\zeta = 1$ means that all information from previous adaptation cycles is ignored. The operator fitness $p_o^{(t+1)}$ is computed from the weighted average $\tilde{p}_o^{(t+1)}$, such that all operator probabilities sum to one and are not lower than the bound $p_{\min} < 1/|\Omega|$. We initialize $\tilde{p}_o^{(0)} = p_o^{(0)}$ for all $o \in \Omega$. If we have no a priori information about the operator performance, we set $p_o^{(0)} = 1/|\Omega|$ for all $o \in \Omega$. Setting $\zeta = 1$ and $\tau = 1$ yields the update rule proposed in [28].

The described algorithm has the following desired properties: [3]

[3] Note that any sequence $a_{n+1} = \zeta x + (1-\zeta)a_n$ converges to x as $n \to \infty$ and $\zeta \in (0,1]$ regardless of the initial value of a_n.

1. For all $o \in \Omega$ and all generations t it holds $p_o^{(t)} \geqslant p_{\min}$.
2. If $o, o' \in \Omega$ and for all generations $t > t_0$
$$\sum_{i=0}^{\tau-1} q_o^{(t-i)} > \sum_{i=0}^{\tau-1} q_{o'}^{(t-i)}$$
then
$$\lim_{t\to\infty} p_o^{(t)} > \lim_{t\to\infty} p_{o'}^{(t)}.$$
3. If $o, o' \in \Omega$ and for all generations $t > t_0$
$$\sum_{i=0}^{\tau-1} q_o^{(t-i)} = \sum_{i=0}^{\tau-1} q_{o'}^{(t-i)}$$
then
$$\lim_{t\to\infty} p_o^{(t)} = \lim_{t\to\infty} p_{o'}^{(t)}.$$

Note that if all operators have performance 0, then the operator fitness values are adapted such that they approach equal operator fitness $1/|\Omega|$. This is in contrast to the algorithm suggested by Davis [18], where in this case the operator probabilities would not be altered.

The proposed adaptation algorithm itself has free parameters, namely $p_{\min}$, τ and ζ. However, in general the number of free parameters is reduced compared to the number of parameters that are adapted. Further, the adaptation adds a new quality to the EA as the operator probabilities can vary over time. Guidelines for the choice of the new parameters exist. The value of τ determines how often the operator fitness values are updated. A larger τ leads to a better estimate of the operator performance, because $\tau \cdot \lambda \cdot p_o^{(t)}$ gives the expectation of the number of times an operator o is sampled between updates (here λ is the number of offspring produced at each generation). The damping parameter ζ, which corresponds to the momentum parameter in some gradient descent algorithms, is used to control fluctuations; decreasing ζ can compensate for small τ. To our experience, the new parameters are very robust.

Here we do not consider dependencies between operators. There might be operators that prepare beneficial variations and operators that work only in conjunction with other operators. However, in order to estimate higher order relations between operators a large sample is needed, i.e., $\tau \cdot \lambda$ has to be large. Davis [17,18] proposed a method where the development of each individual is stored and credit can be passed back to preceding variations. Such a mechanism could be added to our algorithm.

Our method is similar to the derandomized adaptation of the Gaussian mutation distribution in evolution strategies as proposed in [25]. The adaptation takes place at the population level and is based on effects of the operators in the fitness space. Further, the adaptation is derandomized in the sense that adaptation is deterministic. The damping effect of the momentum term can be compared to the evolution path considered in [25]. However, there is a major difference between the adaptation of operator probabilities in discrete optimization and adaptation of mutation distributions in real-valued ES: In $\mathbb{R}^n$ a search direction exists. Most methods in the area of strategy adaptation—including ours—assume implicitly that something like a search direction exists. For example, adding a particular component once to a chemical plant may be beneficial. But does this constitute something like a direction, i.e., does this imply that adding more of that component is also beneficial, or that adding the same component to other structures represented in the current population is advantageous? In the remainder of this study, we show that such a "direction" may exist in structure optimization tasks. As an example, we consider topology optimization of neural networks.

4. Experimental evaluation

4.1. Structure optimization of neural networks

In this investigation, we concentrate on feed-forward neural networks (NNs). Consider a network $\mathcal{N}$ with d external inputs and m external outputs consisting of n neurons $v_1, \ldots, v_n$. Additionally, there is an extra unit v_0 with constant output for the implementation of bias (threshold) parameters. The topology of the network can be described by a graph $G_{\mathcal{N}} = (V_{\mathcal{N}}, E_{\mathcal{N}})$ with vertices $V_{\mathcal{N}} = \{v_0, \ldots, v_n\}$. There is an edge $(v_j, v_i) \in E_{\mathcal{N}}$ iff the neuron v_i gets input from neuron v_j. A graph representing a NN is regarded as *valid* iff each hidden node (i.e, neither external input nor output) lies on a path from an external input unit to an external output unit and there are no cycles.

In our exemplary topology optimization algorithm, valid networks are encoded using a *direct* encoding, i.e., each connection in the NN is represented explicitly in the genotype together with the corresponding weights.

There are three classes of elemental operators (similar to the algorithm proposed in [6]). First, there are node-altering operators:

addNode. A new node is added, which gets two inputs and one output connection obeying the layer restrictions (i.e., a maximum number of layers).

deleteNode. A hidden node, say v_a, and all connections to or from v_a are deleted. If v_a was the only input to a hidden node, this node is connected with one of the former inputs to v_a, which is chosen randomly. If after the deletion a hidden node has no output connection, this node is connected with one of the former outputs of v_a.

Second, connection-altering operators are used:

addConnection. A forward connection is added obeying the layer restrictions.

deleteConnection. A connection that is not necessary for the network to be valid is deleted.

Third, there is an operator that alters only the weights:

jogWeights. For each weight, a random value is drawn from a Gaussian distribution with zero mean and variance $\sigma^2 = 0.1$ and added to the weight.

In every generation, each parent produces one offspring. Elemental operators are chosen randomly and are applied to the offspring. The process of choosing and applying an operator is repeated $1 + x$ times, where x is the realization of a Poisson distributed random number with mean μ. This procedure inspired by [15] allows for "correlated" mutations and ensures detection of the optimal structure with probability one assuming a finite structure space [44]. In all our experiments, setting $\mu = 1$ yielded significantly better results than $\mu = 0$. Hence, in the following only the results for $\mu = 1$ are reported.

After mutation, each individual is trained by gradient-based optimization. This learning phase is stopped either after 100 cycles or when the *training progress* as defined in [40] measured after a *training strip* of length 5 dropped below 0.01 (an algorithm to adapt the number of training cycles during structure optimization is presented in [29]). An improved version of the Rprop algorithm is used for training [43,31,32]. We use Lamarckian evolution, i.e., inheritance of acquired characteristics [14]: after the learning process the weights are stored in the genotype.

The number of hidden layers is restricted to two, i.e., the maximum length of a (directed) path from an external input to an external output is three. EP-style tournament selection with five opponents is applied to determine the parents for the next generation [22].

We would like to emphasize that our investigation focuses on the operator adaptation. Rather than being as efficient as possible, the described algorithm is intended to be instructive. In particular, we preferred to keep the elemental mutation operators simple and to not consider generalization explicitly in our evaluations.

4.2. Test problems

We use two real-world test problems from the PROBEN1 benchmark collection [39], namely the *diabetes*1 and *cancer1* data sets. These are binary classification tasks; a 1-of-2 encoding is used for the output. In the *diabetes*1 classification problem there are 8 inputs. The training set consists of 384 patterns, where some of the patterns contain meaningless zero entries. The *cancer1* problem has 9 inputs and the data set consists of 350 patterns.

The overall error that determines the fitness of a network $\mathcal{N}$ is computed as

$$\mathcal{E}(\mathcal{N}) = v\mathcal{E}_{\text{class}}(\mathcal{N}) + |E_{\mathcal{N}}| + \mathcal{E}_{\text{mse}}(\mathcal{N}). \tag{6}$$

Here, $\mathcal{E}_{\text{class}}$ is the classification error after learning has stopped, i.e., the percentage of wrongly classified patterns, and $\mathcal{E}_{\text{mse}}$ is the mean-squared error. The term $|E_{\mathcal{N}}|$ counts the degrees of freedom (DOF) of the network, i.e., the number of weights and bias parameters. The weighting factor v is set to 10^6 in our experiments, so normally $v\mathcal{E}_{\text{class}}(\mathcal{N}) \gg |E_{\mathcal{N}}| \gg \mathcal{E}_{\text{mse}}(\mathcal{N})$. This fitness function is not arbitrary; there are several reasons to prefer the smaller of two networks with equal classification error. For example, smaller networks can be (re)trained and evaluated faster on sequential hardware, may be easier to analyze, and the costs of their hardware implementation may be lower. Further, there are arguments that smaller networks show better generalization behavior (see [13, Chapters 9,10] for references).

We run the structure optimization algorithm for the test problems without operator adaptation, with adaptation using the benefit, and with adaptation using the absolute benefit. For each setup 64 independent trials are performed using the same 64 initializations for the three different algorithms. We assume that there is no prior knowledge about how to choose the operator fitness (usually, node-altering operators would be applied less often than connection-altering ones) and start each trial with equal operator probabilities, $p_o^{(0)} = 1/|\Omega|$ for all $o \in \Omega$.

In the case of the diabetes task, there are 4–6 hidden neurons in the NNs that form the initial population. In contrast, the initial populations for the cancer problem contain networks with 10–20 hidden neurons. The reason for the different initializations is that we want to observe different adaptation dynamics; the different initial conditions have been assigned arbitrarily to the problems. In all the trials with variable operator probabilities, we use the same parameters for the adaptation algorithm, where the ad hoc values $\tau = 4$, $p_{\min} = 0.1$ and $\zeta = 0.3$ have not been tuned. The population size is set to 20.

To get an idea about the magnitude of the overall error, network topologies that gave good results in previous investigations (an 8–2–2–2 and a 9–4–2–2 feed-forward NN with all shortcut connections for *diabetes*1 and *cancer*1, respectively [39]) have been trained 100 times for 1000 cycles. After *every* training cycle, we calculated the overall error Eq. (6). The best result achieved was 143 296 for the diabetes and 5814 for the cancer task. However, the training aimed only at reducing the mean-squared error and not at optimizing Eq. (6). Therefore, a comparison with the overall error of the evolved networks has a bias.

4.3. Results

The fitness trajectories in Fig. 1 show that the adaptive EAs outperform the static ones. The medians of the final errors of the evolved NNs are much smaller than the best results achieved with the standard architectures. Adaptation based on the benefit, Eq. (2), tends to be better than adaptation using the absolute benefit, Eq. (3). However, the differences are not statistically significant (e.g., Wilcoxon rank sum test[4] in generation 250, $p > 0.1$). In the case of the diabetes task, the EA using the benefit performs statistically better than the static method (e.g., generation 250, $p < 0.005$). The other adaptive algorithm also gives better results than the static one (e.g., generation 250, $p < 0.05$). In the cancer task, the differences between the adaptive EA using the benefit and the method using fixed operator probabilities are statistically significant (e.g., generation 250, $p < 0.05$).

Looking at the trajectories of the operator probabilities averaged over the 64 trials, Figs. 2 and 3, it can be found that the operator fitness values adapt in an intuitive and task-dependent manner and that their ranking changes during the evolutionary process. In the diabetes task, where the initial networks only have few weights, it is beneficial to add new nodes and connections to make it easier for the NNs to classify more patterns correctly. Fig. 2 shows the different adaptation dynamics dependent on whether the benefit or the absolute benefit is used. When using the benefit (upper plot) the operators' probabilities stay rather constant after the initial 75 generations. In

[4] We prefer the median to visualize the error trajectories because of its higher robustness. The results are not normally distributed, so the non-parametric Wilcoxon test is used [58].

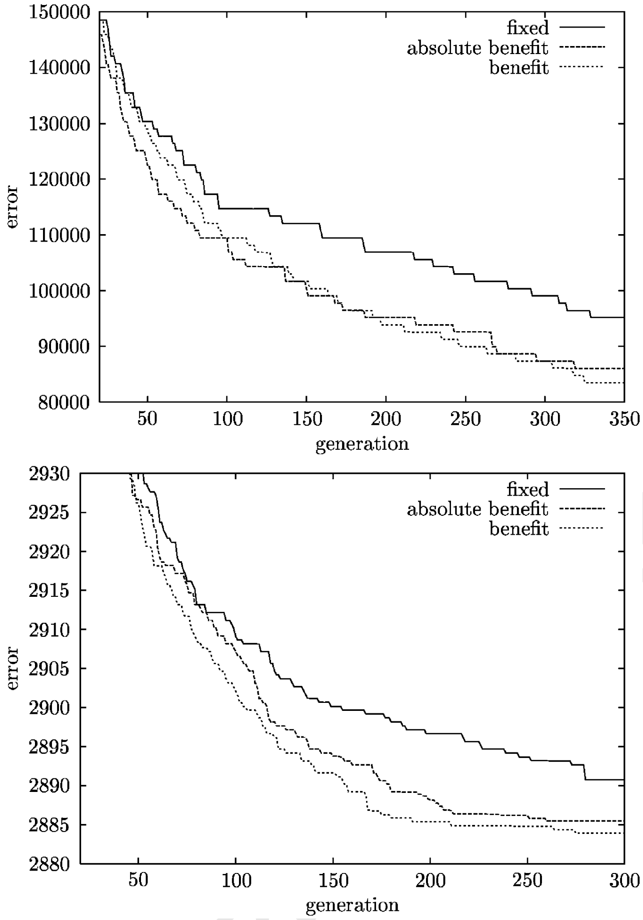

Fig. 1. Medians of the best individual's fitness, based on 64 trials per algorithm. The upper figure corresponds to the diabetes classification task, the lower to the cancer problem.

contrast, when the absolute benefit is used, the operator probabilities approach $1/|\Omega|$. This is because individuals that are fitter than the best individual are generated rarely in the later stages of evolution and therefore all operators are rated similarly.

As the initial populations in the cancer task consisted of comparatively large networks, there is only a small period of about 20 generations in the beginning during which the operators that add DOF are superior. After that, the strategy changes completely and the operators that reduce the complexity perform better and therefore get higher probabilities of application. First, the more drastic *deleteNode* performs best, but after about 75 generations, when the average number of nodes of the NNs in the population has been adapted and smaller changes of the structures are needed, *deleteConnection* becomes more important. This demonstrates the known fact that operators may play different roles at different stages of evolution.

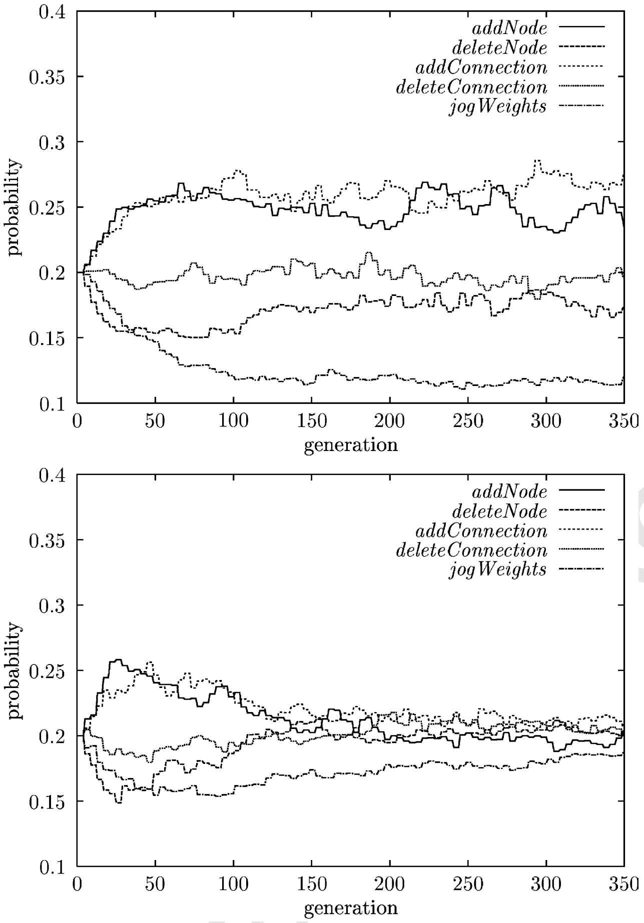

Fig. 2. Operator probabilities for the diabetes task averaged over 64 trials. The upper plot shows adaptation based on the benefit, the lower one based on the absolute benefit.

5. Discussion

In this study, a population level adaptation scheme for operator probabilities is presented, which combines ideas that have been developed in [18,56,25]. The adaptation is deterministic and is based on the fitness improvement induced by the operators. The algorithm has proven to be beneficial when used in structure optimization of neural networks. Our main findings are: structure optimization with operator adaptation performs statistically significantly better than optimization without operator adaptation. The operator probabilities change in an intuitive way during evolution depending on the task and the initialization of the algorithm. Measuring the fitness improvement over the parent tends to give better results than measuring the fitness improvement over the best individual in the population. A nearly optimal static parameter schedule can be

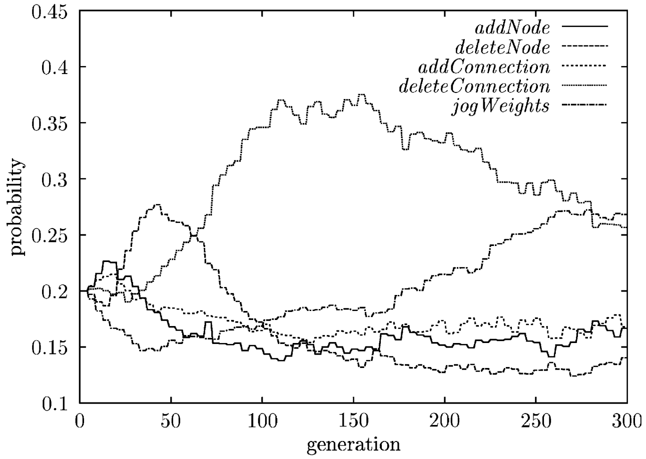

Fig. 3. Operator probabilities based on the benefit for the cancer task averaged over 64 trials.

expected to perform better than any adaptive method, since adaptation takes time to gather the information needed for adjusting the strategy. However, the optimal setup of an EA, which in general changes during the evolutionary process, is usually unknown.

Although fitness improvement measures only "microscopic" behavior, which may have no implication on the "macroscopic" behavior of the EA [57], and the concept of a "search direction" is difficult to apply to structure optimization, our experiments presented here and in earlier work [33] show that the rule of thumb that recent beneficial modifications are likely to be also beneficial for the population in the following generations may still be successful in structure optimization scenarios.

Acknowledgements

We thank Bernhard Sendhoff for his comments on the manuscript. We acknowledge support by the BMBF under grant LOKI, number 01 IB 001 C. Christian Igel acknowledges additional support by the DFG under grant Solesys SE 251/41-1. We thank the John von Neumann Institute for Computing (NIC) in Jülich for the access to their parallel computer system that made the experiments possible.

References

[1] J.T. Alander, An indexed bibliography of genetic algorithms and neural networks, Technical Report 94-1-NN, Department of Information Technology and Production Economics, University of Vaasa, 65101 Vaasa, Finland, 2001.

[2] L. Altenberg, The evolution of evolvability in genetic programming, in: K.E. Kinnear Jr. (Ed.), Advances in Genetic Programming, Chapter 3, MIT Press, Cambridge, MA, 1994, pp. 47–74.

[3] P.J. Angeline, Adaptive and self-adaptive evolutionary computations, in: M. Palaniswami, Y. Attikiouzel, R. Marks, D.B. Fogel, T. Fukuda (Eds.), Computational Intelligence: A Dynamic Systems Perspective, IEEE Press, New York, 1995, pp. 152–163.

[4] P.J. Angeline, Two self-adaptive crossover operations for genetic programming, in: P. Angeline, K. Kinnear (Eds.), Advances in Genetic Programming, Vol. 2, MIT Press, Cambridge, MA, 1996.

[5] P.J. Angeline, D.B. Fogel, L.J. Fogel, A comparison of self-adaptation methods for finite state machines in dynamic environments, in: J. Fogel, P.J. Angeline, T. Bäck (Eds.), Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, 1996, pp. 441–449.

[6] P.J. Angeline, G.M. Saunders, J.B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE Trans. Neural Networks 5 (1) (1994) 54–65.

[7] J. Arabas, Z. Michalewicz, J. Mulawka, GAVAPS—a genetic algorithm with varying population size, in: Proceedings of the First IEEE International Conference on Evolutionary Computation, IEEE Press, New York, 1994, pp. 73–78.

[8] T. Bäck, Self-adaptation in genetic algorithms, in: F.J. Varela, P. Bourgine (Eds.), Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, MIT Press, Cambridge, MA, 1992, pp. 263–271.

[9] T. Bäck, An overview of parameter control methods by self-adaptation in evolutionary algorithms, Fund. Inform. 35 (1–4) (1998) 51–66.

[10] T. Bäck, A.E. Eiben, N.A.L. van der Vaart, An empirical study on GAs "without parameters", in: M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, H.-P. Schwefer (Eds.), Parallel Problem Solving from Nature—PPSN VI, Lecture Notes in Computer Science, Vol. 1917, Springer, Berlin, 2000.

[11] S. Baluja, S. Davies, Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space, in: Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), 1997, pp. 30–38.

[12] H.-G. Beyer, K. Deb, On self-adaptive features in real-parameter evolutionary algorithms, IEEE Trans. Evol. Comput. 5 (3) (2001) 250–270.

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

[14] P.J. Bowler, Lamarckism, in: E.F. Keller, E.A. Lloyd (Eds.), Keywords in Evolutionary Biology, Havard University Press, Cambridge, MA, 1992, pp. 188–193.

[15] K. Chellapilla, Evolving computer programs without subtree crossover, IEEE Trans. Evol. Comput. 1 (3) (1998) 209–216.

[16] K. Chellapilla, D.B. Fogel, Exploring self-adaptive methods to improve the efficiency of generating approximate solutions to traveling salesman problems using evolutionary programming, in: P.J. Angeline, R.G. Reynolds, J.R. McDonnell, R. Eberhart (Eds.), Evolutionary Programming VI: Proceedings of the Sixth International Conference on Evolutionary Programming (EP'97), Lecture Notes in Computer Science, Vol. 1213, Springer, Berlin, 1997, pp. 361–371.

[17] L. Davis, Adapting operator probabilities in genetic algorithms, in: J.D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 1989, pp. 61–69.

[18] L. Davis, Hand book of Genetic Algorithms, Chapter 7, Van Nostrad Reinhold, New York, 1991.

[19] A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Trans. Evol. Comput. 3 (2) (1999) 124–141.

[20] M. Emmerich, M. Grötzner, M. Schütz, Design of evolutionary algorithms: a case study for chemical process networks Evol. Comput. 9 (3) (2001) 329–354.

[21] L.J. Eshelman, J.D. Schaffer, Preventing premature convergence in genetic algorithms by preventing incest, in: R.K. Belew, L.B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91), Morgan Kaufmann, Los Altos, CA, 1991, pp. 10–19.

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

[23] D.B. Fogel, L.J. Fogel, J.W. Atmar, Meta-Evolutionary Programming, in: R.R. Chen (Ed.), Proceedings of 25th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 1991, pp. 540–545.

[24] D.B. Fogel, A. Ghozeil, Using fitness distributions to design more efficient evolutionary computations, in: Proceedings of 1996 IEEE Conference on Evolutionary Computation, Nagoya, IEEE Press, New York, 1996, pp. 11–19.

[25] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2) (2001) 159–195.

[26] G.R. Harik, D. Goldberg, Learning linkage, in: R.K. Belew, M.D. Vose (Eds.), Foundations of Genetic Algorithms 4 (FOGA 4), Morgan Kaufmann, Los Altos, CA, 1997, pp. 247–262.

[27] R. Hinterding, Z. Michalewicz, T. Peachey, Self adaptive genetic algorithm for numeric functions, in: H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature–PPSN IV, Lecture Notes in Computer Science, Vol. 1141, Springer, Berlin, 1996, pp. 420–429.

[28] T.-P. Hong, H.-S. Wang, W.-C. Chen, Simultaneously applying multiple mutation operators in genetic algorithms, J. Heurist. 6 (4) (2000) 439–455.

[29] M. Hüsken, C. Igel, Balancing learning and evolution, in: Genetic and Evolutionary Computation Conference (GECCO'02). Morgan Kaufmann, Los Atlos, CA, 2002.

[30] C. Igel, K. Chellapilla, Fitness distributions: Tools for designing efficient evolutionary computations, in: L. Spector, W.B. Langdon, U.-M. O'Reilly, P.J. Angeline (Eds.), Advances in Genetic Programming, Vol. 3, Chapter 9, MIT Press, Cambridge, MA, 1999, pp. 191–216.

[31] C. Igel, M. Hüsken, Improving the Rprop learning algorithm, in: H. Bothe, R. Rojas (Eds.), Proceedings of the Second International Symposium on Neural Computation (NC2000), ICSC Academic Press, New York, 2000, pp. 115–121.

[32] C. Igel, M. Hüsken, Empirical evaluation of the improved Rprop learning algorithm, Neurocomputing, 2002, in press.

[33] C. Igel, M. Kreutz, Using fitness distributions to improve the evolution of learning structures, in: Congress on Evolutionary Computation (CEC'99), Vol. 3, IEEE Press, New York, 1999, pp. 1902–1909.

[34] B.A. Julstrom, What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm, in: L.J. Eshelman (Ed.), Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95), Morgan Kaufmann, Los Atlos, CA, 1995, pp. 81–87.

[35] H. Mühlenbein, T. Mahnig, A.O. Rodriguez, Schemata, distributions and graphical models in evolutionary optimization, J. Heurist. 5 (2) (1999) 215–247.

[36] J. Niehaus, W. Banzhaf, Adaption of operator probabilities in genetic programming, in: J.F. Miller, M. Tomassini, P.L. Lanzi, C. Ryan, A.G.B. Tettamanzi, W.B. Langdon (Eds.), Genetic Programming, Proceedings of EuroGP'2001, Lecture Notes in Computer Science, Vol. 2038, Springer, Berlin, 2001, pp. 325–336.

[37] A. Ostermeier, A. Gawelczyk, N. Hansen, A derandomized approach to self-adaptation of evolution strategies, Evol. Comput. 2 (4) (1995) 369–380.

[38] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, Linkage problem, distribution estimation, and Bayesian networks, Evol. Comput. 9 (2000) 311–340.

[39] L. Prechelt, PROBEN1—A set of benchmarks and benchmarking rules for neural network training algorithms, Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, 1994.

[40] L. Prechelt, Automatic early stopping using cross validation: quantifying the criteria, Neural Networks 11 (4) (1998) 761–767.

[41] I. Rechenberg, Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution, Werkstatt Bionik und Evolutionstechnik, Frommann-Holzboog, Stuttgart, 1973.

[42] I. Rechenberg, Evolutionsstrategie '94, Werkstatt Bionik und Evolutionstechnik, Frommann-Holzboog, Stuttgart, 1994.

[43] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the RPROP algorithm, in: Proceedings of the IEEE International Conference on Neural Networks, IEEE Press, New York, 1993, pp. 586–591.

[44] G. Rudolph, Convergence Properties of Evolutionary Algorithms, Kovač, Hamburg, 1997.

[45] G. Rudolph, Self-adaptive mutations may lead to premature convergence, IEEE Trans. Evol. Comput. 5 (4) (2001) 410–414.

[46] G. Rudolph, J. Sprave, A cellular genetic algorithm with self-adjusting acceptance threshold, in: Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering: Innovations and Applications, IEE, New York, 1995, pp. 365–372.

[47] C.G. Schaefer, The ARGOT strategy: Adaptive representation genetic optimizer technique, in: J.J. Grefenstette (Ed.), Proceedings of the Second International Conference on Genetic Algorithms and Their Applications (ICGA'87), Lawrence Erlbaum Associates, 1987, pp. 50–55.

[48] J.D. Schaffer, A. Morishima, An adaptive crossover distribution mechanism for genetic algorithms, in: J.J. Grefenstette (Ed.), Proceedings of the Second International Conference on Genetic Algorithms (ICGA'87), Lawrence Erlbaum Associates, 1987, pp. 36–40.

[49] H.P. Schwefel, Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie, Interdisciplinary Systems Research, Vol. 26, Birkhäuser Verlag, Basel, 1977.

[50] H.-P. Schwefel, Evolution and Optimum Seeking, Wiley, Inc., New York, 1995.

[51] J.E. Smith, T.C. Fogarty, Operator and parameter adaptation in genetic algorithms, Soft Comput. 1 (2) (1997) 81–87.

[52] R.E. Smith, E. Smuda, Adaptively resizing populations: Algorithm, analysis and first results Complex Systems 9 (1) (1995) 47–72.

[53] W.M. Spears, Adapting crossover in evolutionary algorithms, in: J.R. Mc-Donnell, R. Reynolds, D.B. Fogel (Eds.), Proceedings of the Fourth Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, 1995, pp. 367–384.

[54] P. Stagge, C. Igel, Neural network structures and isomorphisms: Random walk characteristics of the search space, in: X. Yao, D.B. Fogel (Eds.), IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks (ECNN 2000), IEEE Press, New York, 2000, pp. 82–90.

[55] M. Toussaint, C. Igel, Neutrality: a necessity for self-adaptation, in: IEEE Congress on Evolutionary Computation 2002 (CEC 2002), IEEE Press, New York, 2002.

[56] A. Tuson, P. Ross, Adapting operator settings in genetic algorithms, Evol. Comput. 6 (2) (1998) 161–184.

[57] I. Wegener, On the design and analysis of evolutionary algorithms, in: Workshop on Algorithm Engineering as a New Paradigm, Kyoto University, Japan, 2000, Research Institute for Mathematical Science, 2000, pp. 36–47.

[58] F. Wilcoxon, Individual comparison by ranking methods, Biometr. Bull. 1 (1945) 80–83.

[59] X. Yao, Evolving artificial neural networks, Proc. IEEE 87 (9) (1999) 1423–1447.

[60] A.A. Zhigljavsky, Theory of Global Random Search, Kluwer, Dordrecht, 1991.

Christian Igel received the diploma degree in computer science from the University of Dortmund, Germany, in 1997. He is currently a research staff member with the Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany. His research focuses on biological and technical aspects of neural and evolutionary information processing.

Martin Kreutz was born in Hannover, Germany, on 28 January 1968. He graduated with a diploma in computer Science from the University of Dortmund, Germany, in 1994. In September 1999 he received his doctoral degree in computer science from the University of Dortmund, Germany. From 1994 to 1999 he worked as a research staff member at the Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany. He joined the ZN Vision Technologies AG, Germany, in April 1999 and is currently involved in the development of advanced diagnostic tools for medical applications. His current research interests are in the areas of computer vision, statistics, and evolutionary optimization.