Evolutionary Optimization of Radial Basis Function Classifiers for Data Mining Applications

Oliver Buchtala, Manuel Klimek, and Bernhard Sick, *Member, IEEE*

***Abstract*—In many data mining applications that address classification problems, feature and model selection are considered as key tasks. That is, appropriate input features of the classifier must be selected from a given (and often large) set of possible features and structure parameters of the classifier must be adapted with respect to these features and a given data set. This paper describes an evolutionary algorithm (EA) that performs feature and model selection simultaneously for radial basis function (RBF) classifiers. In order to reduce the optimization effort, various techniques are integrated that accelerate and improve the EA significantly: hybrid training of RBF networks, lazy evaluation, consideration of soft constraints by means of penalty terms, and temperature-based adaptive control of the EA. The feasibility and the benefits of the approach are demonstrated by means of four data mining problems: intrusion detection in computer networks, biometric signature verification, customer acquisition with direct marketing methods, and optimization of chemical production processes. It is shown that, compared to earlier EA-based RBF optimization techniques, the runtime is reduced by up to 99% while error rates are lowered by up to 86%, depending on the application. The algorithm is independent of specific applications so that many ideas and solutions can be transferred to other classifier paradigms.**

***Index Terms*—Data mining, evolutionary algorithm (EA), feature selection, model selection, radial basis function (RBF) network.**

I. Introduction

TWO key tasks must be mastered in many data mining applications addressing classification problems: The first is the selection of features (attributes) from a given, possibly large set of possible features (so-called *feature selection*). The second is the optimization of the classifier's structure with respect to the features selected (so-called *model selection*). It is obvious that the two problems should be addressed simultaneously to achieve the best classification results.

In this article we start with the following assumptions.

1) *Radial basis function networks* (RBF, [1]–[4]) are used for classification. Here, these neural networks are trained to estimate posterior probabilities of class membership by means of mixtures of Gaussian basis functions and hyperplanes. From a structural viewpoint, RBF networks are closely related to direct kernel methods [5] and support vector machines (SVM) with Gaussian kernel functions [1], [6].
2) *Evolutionary algorithms* (EA, [7], [8]) are used for architecture optimization (combined feature and model selection) of the RBF networks. Here, this class of optimization algorithms is chosen because the search space is high-dimensional and the objective function is noisy, deceptive, multimodal, and nondifferentiable.

Evolutionary optimization of RBF architectures is in no way a new idea, but existing approaches typically suffer from the problems of a high runtime and a premature convergence in local minima. Often, these two problems are closely related: Due to a high runtime smaller populations with lower diversity are evolved and convergence in a local minimum becomes more likely. Hence, we take over the best from existing approaches—such as standard encoding schemes for the representation of individuals or standard recombination and mutation operators—and we integrate various techniques that reduce the runtime of the EA radically. In particular, we use methods for:

1) fast fitness evaluation of individuals (hybrid training of RBF networks, lazy evaluation);
2) consideration of soft constraints by means of penalty terms (e.g., to prefer smaller network structures);
3) adaptive control of the evolutionary optimization process by means of a temperature coefficient.

Due to a significantly reduced runtime and a goal-oriented search more and fitter solutions can be evaluated within shorter time. Therefore, it can be expected that better solutions with higher classification rates can be obtained.

Altogether, an algorithm is introduced that can be utilized for a wide variety of data mining tasks to obtain and to apply new, domain-specific knowledge. This approach efficiently instantiates already known techniques as well as innovative, novel ideas. Its advantages are outlined by means of four real-world data mining applications: Intrusion detection in computer networks, biometric signature verification, customer acquisition with direct marketing methods, and optimization of chemical production processes. Compared to some of our earlier work on EA-based RBF optimization, the runtime is reduced by up to 99% and the error rates are decreased by up to 86%, depending on the application.

The remainder of this article is structured as follows. First, the state of the art is analyzed to motivate our work (Section II) and the RBF networks used are described (Section III). Then, the EA for architecture optimization is introduced—with a strong focus on the innovative aspects mentioned above (Section IV). After that, the advantages of the approach are set out by means of various application examples (Section V). Finally, the main

Manuscript received June 8, 2004; revised September 21, 2004. This paper was recommended by Associate Editor M. Berthold.

O. Buchtala and B. Sick are with the Faculty for Computer Science and Mathematics, University of Passau, 94032 Passau, Germany (e-mail: buchtala@fmi.uni-passau.de; sick@fmi.uni-passau.de).

M. Klimek is at Angerstrasse 12, 85354 Freising, Germany (e-mail: klimek@box4.net).

Digital Object Identifier 10.1109/TSMCB.2005.847743

findings are summarized and an outlook on future work is given (Section VI).

II. State of the Art

In this section, the state of the art concerning evolutionary optimization of RBF networks is investigated. The results of this survey will motivate a new approach.

A. Related Work

This article focuses on "feature selection" and "model selection" for RBF networks. Potential feature selection algorithms are described in [9], [10]. In general, *filter* and *wrapper* approaches can be distinguished. The problem of model selection for neural networks is discussed in [11]–[13] in greater detail. Usually, these techniques are categorized as being either *constructive* (growing techniques), *destructive* (pruning techniques), or *hybrid*. The subject of data mining and knowledge discovery with EA is addressed in [14].

The combination of EA and neural networks in general is investigated in [15] and [16]. Here, we discuss examples of the combination of EA and RBF networks. Altogether, we investigated 64 publications ([17]–[80]) where EA or closely related techniques (genetic algorithms, evolution strategies, or immunity-based approaches, for instance) are applied to optimize RBF networks or closely related paradigms in some respect. Related paradigms are hyper basis function networks [21], probabilistic neural networks [22], [29], [31], [47], second-order multilayer perceptrons (MLP) [36], hybrid RBF-MLP networks [25], [49], Volterra polynomial basis function networks [40], [52], projection neural networks [57], RBF networks with dynamic receptive fields [59], beta basis function neural networks [65], [66], functional link networks [38], or a neuro-fuzzy controller based on RBF networks [51].

1) The *computation of weights* (particularly centers and radii of basis functions and/or output weights) by EA is suggested in the majority of the publications. The centers are optimized in [17]–[20], [23], [28]–[33], [37]–[42], [45], [46], [48]–[55], [57], [58], [62], [63], [65]–[71], and [73]–[79], for instance. Radii are considered in [17], [19]–[23], [26]–[28], [30], [32], [38], [45], [46], [50], [51], [54], [56], [58], [60], [62], [63], [65]–[69], [71], and [74]–[79], and weights in the output layer in [18], [19], [51], [56], [57], [60], and [61]. Particular parameters of basis function types (e.g., shaping parameters) are optimized in [57], [59], [65], and [66]. Weights that are not optimized by an EA are adjusted by means of various other techniques such as k-means clustering, nearest neighbor techniques, Learning Vector Quantization, Cholesky decomposition, singular value decomposition, orthogonal least squares, quasi-Newton techniques, Levenberg–Marquardt training, etc.
2) Closely related to the weight determination problem is the problem of *structure* or *architecture* evolution. Architecture parameters (apart from features, see below) are optimized in about half of the publications. The number of centers (hidden neurons) is considered in [17]–[20], [25], [29]–[31], [33], [40], [42], [45], [46], [50], [52], [54], [57], [62], [67], [68], [70], [71], [73], [75], and [77], for instance. Other architecture parameters are the type of basis functions [57], [58], the training time (epoch number) [36], and parameters of training algorithms [26], [27], [36], [63]. Examples of such parameters are learning rate and momentum or a regularization parameter used for training with regularized orthogonal least squares training.
3) *Feature selection* for RBF networks by means of EA is investigated in only five of the publications: [34] and [35] describe class-dependent feature selection by masking of features; other examples are [47], [48], and [72].
4) *Rule extraction* from trained networks is shown in [43].
5) The *combination of networks* in form of ensembles by means of EA is conducted in [24], [44], and [64].

In our earlier work, we described an EA-based method for feature selection which also optimizes the number of hidden nodes in RBF networks as well as some other architecture parameters [81]–[83]. This approach has successfully been applied to two problems in the areas of *intrusion detection (ID)* in computer networks and *direct marketing (DM)*.

In the following, we will summarize the most important contributions of the mentioned publications.

Individuals in an evolutionary approach are either complete networks or basis functions (sets or single functions) which constitute a network. The latter idea is investigated in [23] where a cooperative-competitive evolution of centers and radii is proposed. The approach leads to moderate computation times but requires a fitness function that promotes competition among similar centers and cooperation among different centers at the same time ("niche creation"). Similar methods are introduced in [28], [67], and [69]. Also, [77] describes an approach that takes cooperative-competitive evolution into account. The fitness function combines concepts such as cooperation, speciation, and niching. The probability for choosing a reproduction operator is determined by means of a Mamdani type fuzzy inference system.

Most methodologies use direct encoding schemes (values of centers and radii, for instance) but there are some articles that propose indirect encoding techniques, e.g., grammars or functions that generate networks. This alternative solution helps to reduce the temporal effort. In [63], locations of basis functions are governed by space-filling curves whose parameters are evolved. Basis function centers are generated at equidistant points along this curve. In [42] and [46], a cluster distance factor is the only parameter to be evolved. This parameter—defined by the maximum distance between an input sample and a center—allows the number of basis functions to increase iteratively.

Specialized mutation and crossover operators (operators that exploit the particular properties of RBF networks) are used in many approaches. Only [62], however, investigates the application of these operators in detail by comparing the effects of various operators. Examples are the pruning of networks by ranking basis functions and the increase of the number of basis functions in poorly modeled regions of the input space. Also, this publication compares single criterion and multicriteria approaches of

evolution. Other, important articles on multiobjective optimization are [40], [52], and [70].

The integration of advanced learning approaches into the evolution process is mentioned in the following articles: In [22], the recursive orthogonal least-squares algorithm is applied to optimize the centers (number and locations). The regularized orthogonal least-squares algorithm (an efficient forward subset selection algorithm) is used in [26] and [27]. A regularization parameter is optimized by the EA.

Finally, [68] should be mentioned as this publication provides a good overview of related work.

B. Motivation for a New Approach

An assessment of related work in regard to real-world data mining applications is difficult. Data mining tasks with millions of patterns and hundreds of features are not uncommon. Most of the 64 publications (about 70%) utilize very small and often simple benchmark data sets (e.g., two spirals, trigonometric functions, polynomials, iris, glass, or sunspot). Real-world data is considered in 19 publications, e.g., from robotic applications [19], [24], [44], [64], medical or biometrical applications [39], [49], power systems modeling and control [46], [47], [61], or chemical applications [60]. Typically, these publications focus on one particular application problem. Hence, the universality of methods does not become clear. Besides that, most of the publications do not set out the repeatability of results even if a stochastic optimization algorithm is applied. Repeated runs are conducted in 12 of the 64 publications only, namely in [17], [23], [28], [38], [54], [59], [62], [71], [73], [76], [78], and [79].

Altogether, two general drawbacks of many existing approaches can be noticed.

- *Coding (representation) scheme:* Often, binary representations are used for integer or real parameters. If reproduction operators such as one-point crossover (recombination) or bit switch (mutation) are applied, effects such as positional bias are usually neglected. For some of the representation schemes used, the reproduction operators do not guarantee that the descendants describe valid solutions and repair mechanisms are needed. On the other hand, complex and solution-specific representation schemes and operators are defined in some cases.
- *Runtime:* The runtime of the optimization process is mentioned (but not investigated) in only a few publications. However, a long runtime seems to be a major problem of many evolutionary approaches. With existing approaches applied to real-world data mining problems, the user would have to accept a runtime of up to several days or he would have to risk a premature convergence in local minima. The former is definitely not practicable in many applications. The latter is due to small populations yielding a loss of diversity.

Whereas the first problem can be avoided using standard techniques for the representation of individuals together with well-known operators for recombination and mutation, the second problem can be seen as a substantial challenge. We claim that the runtime of existing approaches must be reduced by more than 95% in order to make them applicable to a wide range of practical data mining problems. At the same time, premature convergence or overfitting to a particular data set must be avoided, and any solution (point in the search space) must be within reach. If more different solutions could be investigated within shorter time, it can also be expected that better solutions can be achieved. Evolutionary optimization of weights (centers, radii, and output weights) should not be taken into consideration (cf. [80]) because various fast and efficient training methods exist, in particular for classification problems (see below).

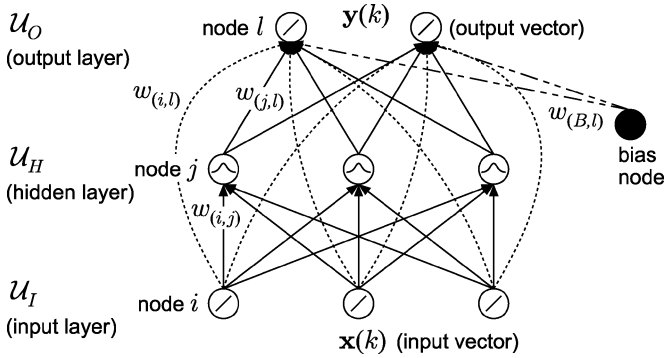

Fig. 1. Example of an RBF network.

III. Classification With Radial Basis Function Networks

Radial basis function (RBF) networks combine a number of different concepts from approximation theory, clustering, and neural network theory [1], [2], [11]. A key advantage of RBF networks for practitioners is the clear and understandable interpretation of the functionality of basis functions. Also, fuzzy rules may be extracted from RBF networks (cf. [84]) for deployment in an expert system.

The RBF networks used here may be defined as follows (see Fig. 1) [13].

1) RBF networks have three layers of nodes: input layer $\mathcal{U}_I$, hidden layer $\mathcal{U}_H$, and output layer $\mathcal{U}_O$.
2) Feed-forward connections exist between input and hidden layers, between input and output layers (shortcut connections), and between hidden and output layers. Additionally, there are connections between a bias node and each output node. A scalar weight $w_{(i,j)}$ is associated with the connection between nodes i and j.
3) The activation of each input node (fanout) $i \in \mathcal{U}_I$ is equal to its external input

$$a_i(k) \stackrel{def}{=} x_i(k)$$

where $x_i(k)$ is the ith element of the external input vector (pattern) $\mathbf{x}(k)$ of the network ($k = 1, 2, \ldots$ denotes the number of the pattern).
4) Each hidden node (neuron) $j \in \mathcal{U}_H$ determines the Euclidean distance between "its own" weight vector $\mathbf{w}_j \stackrel{def}{=} (w_{(1,j)}, \ldots, w_{(|\mathcal{U}_I|,j)})^T$ and the activations of the input nodes, i.e., the external input vector

$$s_j(k) \stackrel{def}{=} ||\mathbf{w}_j - \mathbf{x}(k)||.$$

The distance $s_j(k)$ is used as an input of a radial basis function in order to determine the activation $a_j(k)$ of node j. Here, Gaussian functions are employed

$$a_j(k) \stackrel{def}{=} e^{\left(-s_j(k)^2/r_j^2\right)}.$$

The parameter r_j of node j is the radius of the basis function; the vector $\mathbf{w}_j$ is its center.

Any other function which satisfies the conditions derived from theorems of Schoenberg or Micchelli described in [2] may also be used as a basis function. Localized basis functions such as the *Gaussian* or the *inverse multiquadric* are usually preferred.

5) Each output node (neuron) $l \in \mathcal{U}_O$ computes its activation as a weighted sum

$$a_l(k) \stackrel{def}{=} \sum_{j=1}^{|\mathcal{U}_H|} w_{(j,l)} \cdot a_j(k) + \sum_{i=1}^{|\mathcal{U}_I|} w_{(i,l)} \cdot a_i(k) + w_{(B,l)}.$$

The external output vector of the network, $\mathbf{y}(k)$, consists of the activations of output nodes, i.e., $y_l(k) \stackrel{def}{=} a_l(k)$.

The activation of a hidden node is high if the current input vector of the network is "similar" (depending on the value of the radius) to the center of its basis function. The center of a basis function can, therefore, be regarded as a prototype of a hyperspherical cluster in the input space of the network. The radius of the cluster is given by the value of the radius parameter.

In the literature, some variants of this network structure can be found, some of which do not contain shortcut connections or bias neurons.

Parameters (centers, radii, and weights) of the RBF networks must be determined by means of a set of training patterns $\mathcal{L} \stackrel{def}{=} \{(\mathbf{x}(k), \mathbf{t}(k))\}$ with a a target vector $\mathbf{t}(k) \stackrel{def}{=} \left(t_1(k), \ldots, t_{|\mathcal{U}_O|}(k)\right)^T$ and $|\mathcal{L}| \geq |\mathcal{U}_H|$ (supervised training). For a given input $\mathbf{x}(k)$ the network is expected to produce an external output $\mathbf{t}(k)$. The least-squares error (LSE) function will be applied here to assess the difference between $\mathbf{t}(k)$ and $\mathbf{y}(k)$. Typical training algorithms used to determine the parameters are either gradient-based techniques (such as Backpropagation, Resilient Propagation, or Quickprop) or clustering techniques in combination with methods for the solution of linear least-squares problems (e.g., k-means and singular value decomposition). An overview of various training methods for RBF networks is given in [85]–[87].

For a classification problem with a set of classes $\mathcal{C}$, each class $c_l \in \mathcal{C}$ is typically assigned its own output neuron (i.e., $|\mathcal{C}| = |\mathcal{U}_O|$ and $l \in \{1, \ldots, |\mathcal{U}_O|\}$). For training purposes, an orthogonal representation of classes is used at the output nodes (1-of-$|\mathcal{C}|$ representation). That is, if the external input vector $\mathbf{x}(k)$ belongs to class $c_l \in \mathcal{C}$, the elements of the target vector are $t_i(k) = 1$ for $i = l$ and $t_i(k) = 0$ otherwise.

After training, the actual output $y_l(k)$ of node $l \in \mathcal{U}_O$ for a given input vector $\mathbf{x}(k)$ is interpreted as the posterior probability $p(c_l|\mathbf{x}(k))$ of class membership. That is, the network training is regarded as a mixture density estimation problem. A winner-takes-all approach for the final decision about class membership minimizes the probability of misclassification [11]. That is, for an external input vector $\mathbf{x}(k)$ the class predicted by the network is $c_{l'} \in \mathcal{C}$ with

$$l' = \arg\max_{l \in \mathcal{U}_O}(y_l(k)).$$

In the following, let $ip(\mathbf{t}(k)) \in \mathcal{C}$ and $ip(\mathbf{y}(k)) \in \mathcal{C}$ (*ip*: interpretation function) be the actual and the predicted class for an input vector $\mathbf{x}(k)$. This class information can be obtained by means of the definition of the representation of the target vector $\mathbf{t}(k)$ and the winner-takes-all approach applied to the external output vector $\mathbf{y}(k)$, respectively.

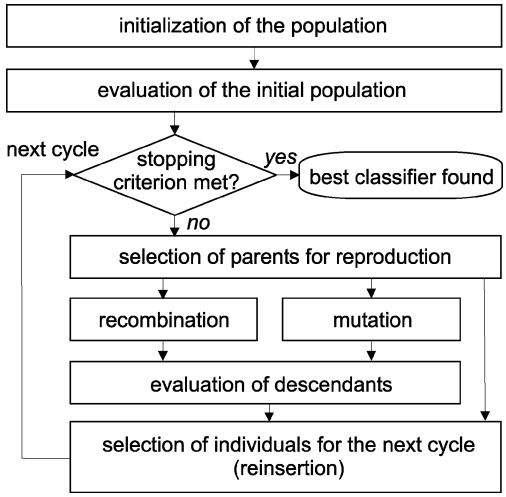

Fig. 2. Application flow of the evolutionary algorithm.

IV. Evolutionary Optimization of RBF Networks

In this section, a schematic overview of the EA used for RBF optimization is given. Then, the applied standard techniques are sketched and our innovative extensions are described in detail. From the viewpoint of model and feature selection, this approach can be characterized as a hybrid wrapper technique.

A. Overview

From an algorithmic perspective, evolution is a stochastic search strategy that can be used to solve a wide range of optimization tasks including architecture optimization of neural networks [7], [8].

Fig. 2 sets out the application flow of the EA for architecture optimization of RBF networks. A population is a set of individuals (i.e., a set of solutions) of fixed size. Individuals are RBF networks. The evolutionary operators for recombination and mutation work on a representation of the individual, the so-called *genotype*. Selection for reproduction or for reinsertion is based on an individual's fitness which is evaluated with respect to an objective function. The fitness describes the behavior of the individual in its environment, the so-called *phenotype*.

Here, the genotype is an abstract representation of the network architecture that is subject to optimization and the phenotype is a trained network. Specific challenges in the case of architecture optimization arise due to nondeterminism: Patterns for training, validation, and testing are selected randomly, the order of patterns in the training data set is chosen randomly,

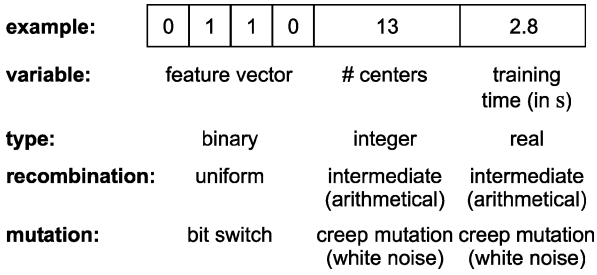

Fig. 3. Representation scheme and reproduction operators.

the training of networks may start from a random parameter initialization, etc. As a consequence, different phenotypes may emerge from the same genotype, and a particular phenotype may be produced from different genotypes (cf. the *competing conventions* problem mentioned in [78]–[80]). For example, the choice of another data set for training may lead to a different classification rate, and the same classification rate may be achieved with two different epoch numbers taken for network training. That is, a wide range of uncertainties must be met appropriately.

B. Standard Components of the Evolutionary Algorithm

The development of the new approach was guided by the idea that well-known evolutionary techniques should be applied as far as possible. To keep the runtime of the EA short, only the most important architecture parameters are optimized.

1) Representation of Individuals: Deciding on the representation of an individual is a pivotal step as the representation interacts highly with the choice of evolutionary operators. Thus, the representation has a commensurate impact on the success of an EA [88]. Here, the genotype of an individual is represented by the following.

1) **The feature vector:** This vector is a binary vector, where each bit indicates the presence ("1") or absence ("0") of one of the possible features in the currently selected feature subset.
2) **The number of centers:** This number (which is equivalent to the number of hidden neurons) is represented by an integer.
3) **The training time:** This parameter (represented by a real, positive scalar) restricts the total runtime available for the construction and evaluation of one individual.

This "mixed" representation scheme (see Fig. 3) requires the application of different reproduction operators for the different parts of the genotype (variables).

The representation that does not encode details of a network architecture—such as the existence of single connections or the assignment of a feature to a specific input node—can be seen as a *high-level specification scheme* (*weak encoding scheme*) [89]–[91]. An overview of various neural network encoding strategies is provided in [92].

2) Initialization of the Population: The feature vector and the number of hidden nodes of individuals in the initial population are initialized randomly. Values of user-defined parameters are taken into account, e.g., the expected number of features and centers or standard deviations for these variables. The training time is identical for all individuals in the initial population (user-defined).

3) Evaluation of Individuals (Initial Population or Descendants): Individuals are evaluated by means of a fitness function. A *basis fitness* $fit^{(basis)}$ is calculated with respect to the application problem at hand using a set of validation patterns $\mathcal{V} \stackrel{def}{=} \{(\mathbf{x}(k), \mathbf{t}(k))\}$ with $\mathcal{L} \cap \mathcal{V} = \emptyset$. Here, the sets $\mathcal{L}$ and $\mathcal{V}$ vary in each cycle (see Section IV-C1b).

For *classification problems*, the *classification rate* is calculated assuming an equal a-priori distribution of classes. That means, the loss of correct classification is zero, whereas the loss of errors for each class is proportional to the a-priori probability of that class [93]. Hence, the classification rate of a certain network with respect to a validation set $\mathcal{V}$ is

$$class_{\mathcal{V}} \stackrel{def}{=} \frac{1}{|\mathcal{C}|} \cdot \sum_{c_l \in \mathcal{C}} \frac{|\{k \in \{1,\ldots,|\mathcal{V}|\} \,|\, ip(\mathbf{t}(k)) = c_l \wedge ip(\mathbf{y}(k)) = c_l\}|}{|\{k \in \{1,\ldots,|\mathcal{V}|\} \,|\, ip(\mathbf{t}(k)) = c_l\}|}.$$

In this case, the basis fitness is $fit^{(basis)} \stackrel{def}{=} class_{\mathcal{V}}$.

In a *sample subset selection problem* appropriate input patterns must be selected from a set of input patterns. The real-valued output of a trained network is evaluated ("scored") in order to select input patterns which are highly correlated with a certain class. Patterns yielding lower scores are assumed to be more or less "irrelevant" to the application problem. The threshold for the higher, more interesting percentile of scores is defined by a so-called *cut-off point*. The *lift factor* is then used to assess the suitability of an individual (network) for an application [94], [95]. Here, the lift factor is defined by the quotient of the response rate of the trained network for a certain class and the percentage of input patterns associated with this class in the overall set of samples. The response rate is the rate of correct classifications which can be achieved using a subset of input patterns selected with respect to the cut-off point. That is, for a validation data set $\mathcal{V}$, a class $c_l \in \mathcal{C}$, and a cut-off point $cut \in (0,1)$ we define $y^{cut} \in \mathbb{R}$ such that

$$\mathcal{V}^{cut} \stackrel{def}{=} \{(\mathbf{x}(k), \mathbf{t}(k)) \in \mathcal{V} \,|\, \mathrm{score}_l(\mathbf{y}(k)) \geq y^{cut}\}$$

with $|\mathcal{V}| \cdot cut \leq |\mathcal{V}^{cut}| < |\mathcal{V}| \cdot cut + 1$. The simplest score function would be $\mathrm{score}_l(\mathbf{y}(k)) \stackrel{def}{=} y_l(k)$. Here, a score function that prefers clear decisions is applied

$$\mathrm{score}_l(\mathbf{y}(k)) \stackrel{def}{=} y_l(k) \cdot \left(y_l(k) - \frac{1}{|\mathcal{U}_O| - 1} \sum_{\substack{l' \in \mathcal{U}_O \\ l' \neq l}} y_{l'}(k) \right).$$

Then, the lift factor of a network is defined by

$$lift^{cut}_{\mathcal{V},c_l} \stackrel{def}{=} \frac{|\{(\mathbf{x}(k), \mathbf{t}(k)) \in \mathcal{V}^{cut} \,|\, ip(\mathbf{t}(k)) = c_l\}|}{|\{(\mathbf{x}(k), \mathbf{t}(k)) \in \mathcal{V} \,|\, ip(\mathbf{t}(k)) = c_l\}| \cdot cut}$$

and the basis fitness is $fit^{(basis)} \stackrel{def}{=} lift^{cut}_{\mathcal{V},c_l}$.

4) Selection for Reproduction: Two selection mechanisms are used here in combination (see Section IV-C3).

- *Stochastic universal sampling* (SUS) [96] prefers fitter individuals but also gives a chance to worse individuals.
- With *elitist selection*, only the fittest individuals are taken.

This approach effects a suitable compromise between fast convergence (elitist selection) and avoidance of a loss of diversity (SUS).

A distance-based approach was chosen for *pairing*, i.e., the selection of parents for recombination: An assessment measure reflecting the dissimilarity of two individuals is calculated for all possible pairs of parents (cf. the *diversity control* mentioned in [44]). This measure is defined by the distance (with respect to the 1-norm) of the feature vectors and the numbers of hidden neurons. This gives a higher priority to the number of hidden neurons compared to a single input feature. Then, the two parents with the highest dissimilarity are taken and the method restarts for the remaining individuals. This approach prevents *inbreeding*, i.e., mating of very similar genotypes.

5) Recombination: Recombination produces descendants of a set of parents by combining genotypes. The two recombination operators used here are *uniform crossover* and *intermediate recombination*. Different operators are applied to different parts of the representation (i.e., variables).

Uniform crossover [7], [96] treats each parameter individually. A descendant inherits the parameter value—with equal probability—from one of the two parents.

Intermediate recombination [96] (or arithmetical crossover [8]) produces descendants in the neighborhood of the parents

$$param_{desc} \stackrel{def}{=} \lambda \cdot param_{parent1} + (1-\lambda) \cdot param_{parent2}$$

where the weight λ is a realization of a random variable with a uniform distribution in $[-h; 1+h]$ and $h \in \mathbb{R}^+$ is a constant, e.g., $h = 0.25$ [96]. The parameter h is used to ensure that the hypercube marking the subspace that contains the representations of all possible descendants does not become smaller in each evolution cycle.

Uniform crossover is used for the feature vector; intermediate recombination is applied to the number of centers (with rounding) and the training time.

6) Mutation: Mutation is a stochastic variation of the genotypes of individuals.

For the feature vector, mutation means the random change of a bit (*bit switch*). The mutation technique used here avoids a drift toward an identical number of zeros and ones in the feature vector. Convergence to a certain number of features is driven by the selection mechanism. If $\mathcal{F}$ is the set and $|\mathcal{F}|$ the number of possible features, then $|\mathcal{U}_I|$ is the number of features actually used by an individual and $|\mathcal{F}| - |\mathcal{U}_I|$ is the number of features not used. For a binary entry $b \in \mathbb{B}$ in the feature vector of an individual the mutation is defined by

$$mut(b) \stackrel{def}{=} \begin{cases} \neg b, & \text{for } z < mp_b \\ b, & \text{otherwise} \end{cases}$$

with z being the realization of a random variable with a uniform distribution in [0, 1] and

$$mp_0 \stackrel{def}{=} \frac{\mu_{switch}}{|\mathcal{F}| - |\mathcal{U}_I|} \text{ and } mp_1 \stackrel{def}{=} \frac{\mu_{switch}}{|\mathcal{U}_I|}$$

where $0 \leq \mu_{switch} \leq |\mathcal{U}_I|$ is a user-defined parameter. The mutation parameters mp_0 and mp_1 are not needed for $|\mathcal{F}| - |\mathcal{U}_I| = 0$ and $|\mathcal{U}_I| = 0$, respectively.

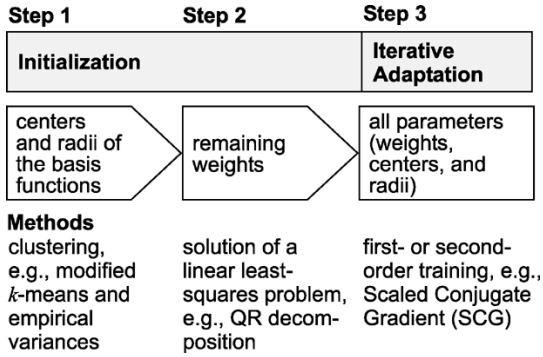

Fig. 4. Training of parameters in RBF networks.

The training time is mutated by adding discrete white noise with an expectation of zero and a certain standard deviation σ, i.e., $\mathcal{N}(0, \sigma)$. A similar technique is used for the number of hidden neurons (with rounding). This kind of mutation is known as *creep mutation* [23], [45], [50], [78], [79].

7) Selection for Reinsertion: After a reproduction step, the descendants must be evaluated (see above) and the population for the next cycle is chosen from all individuals of the current population (including parents and descendants). We apply the elitist selection method in order to keep the number of individuals constant at the beginning of each cycle.

8) Stopping Criterion: For the sake of simplicity, a number of EA cycles must be specified by the user.

C. Innovative Components of the Evolutionary Algorithm

In this section, the innovative extensions to our approach are described. Their development was guided by the idea that the runtime of the EA must be reduced significantly, so that it can be deployed in real-world applications. An additional measure already described is the use of a high-level representation scheme—only the most important architecture parameters are subject to evolutionary optimization.

1) Fast Evaluation: The starting point for runtime reduction is to decrease the time required to evaluate an individual's fitness. There is time needed for the training of a network (i.e., to assign a phenotype to a given genotype). Furthermore, the application of methods that prevent overfitting to a specific data set (such as cross-validation or bootstrapping) also contribute to the time required for the evaluation.

a) Hybrid RBF training: RBF networks allow the application of a specific training concept that consists of several initialization and iterative weight adaptation steps (Fig. 4). We described this concept in detail in [85], [86]. Therefore, only an overview of the three steps is given here:

1) A clustering algorithm is employed for the initialization of the centers and the radii of the basis functions. In substance, this algorithm is a modified k-means approach. The centers of the basis functions correspond to the prototypes of clusters in the input space and the values of the radii depend on the average Euclidean distance between the patterns assigned to the cluster and the cluster centers (cf. definition of empirical variance). The original

k-means algorithm (cf. [97] and [98]) is modified in the following way.

- A stable initialization of prototypes prior to the first k-means step leads to better results and a deterministic behavior of this clustering algorithm. In essence, initial prototypes are chosen with a selection method that aims at maximizing their average Euclidean distance.
- The restriction that centers of the RBF network must correspond to patterns in the training set guarantees that the solution of the linear least-squares problem, that has to be solved in Step 2, exists (see below).
- The clustering may be conducted separately for the patterns of each class which yields a better behavior when classes are difficult to separate ("class-based clustering"). Like the original k-means, the modified k-means is globally convergent.

2) Once the values of centers and radii are found, the remaining parameters—weights of connections between input and output layers, weights of bias connections, and weights between hidden and output layers—are determined in a single step, solving a linear least-squares problem. The values of these weights are then optimal (in a least-squares sense) with respect to the selected values of the centers and radii.

   In practice, the pseudo-inverse that defines the solution of the least-squares problem is not computed explicitly, but the least-squares problem is solved by means of an efficient, numerically stable algorithm (cf. [99] and [100]) such as QR decomposition. If Gaussian basis functions are used (for other permissible functions see [2]) and the centers are chosen to be a subset of the training data and distinct, the pseudo-inverse which describes the solution does exist [2]. These conditions are met here (Step 1); they are sufficient, but not necessary.

3) Finally, an iterative learning algorithm utilizes the current state of the network (i.e., the current values of all parameters that result from Steps 1 and 2) as a starting point for further weight optimization. Here, the scaled conjugate gradient (SCG) learning algorithm is applied [101]. Basically, SCG is a very fast and efficient combination of a Conjugate Gradient algorithm and a Model Trust Region approach (cf. [11] and [102]). Also, any first-order (such as Backpropagation or Resilient Propagation) or second-order algorithm (such as conjugate gradient techniques) may be applied.

A similar training concept (three-phase learning) is advocated in [87]. The training concept offers two vast advantages.

- The overall algorithm converges extremely fast. Typically, the initialization (Steps 1 and 2) already provides very good results.
- The training time available for the iterative adaptation in Step 3 may be adjusted on-line in order to effect a compromise between coarse and fine tuning of the evolutionary search.

Here, the training time of an individual (as given in its representation, see Section IV-B1) is interpreted as follows: Steps 1 and 2 are always executed, Step 3 is iterated until the training time is exceeded.

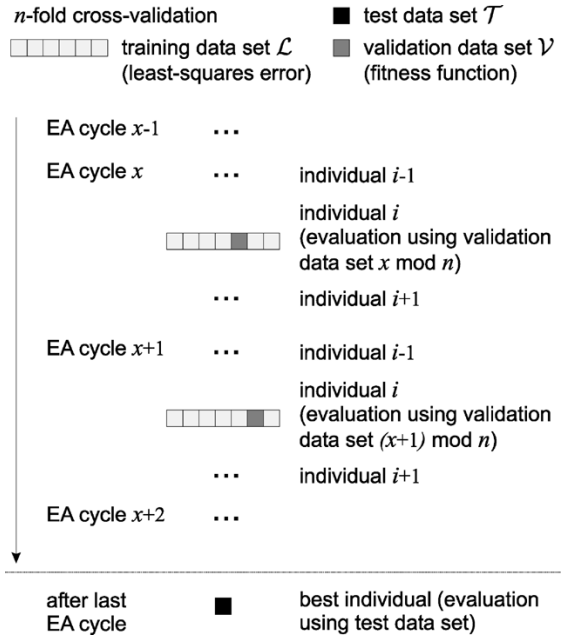

Fig. 5. Lazy evaluation of individuals.

In the 64 related publications (see Section II), a comparable, hybrid training scheme cannot be found; the training time is evolutionary optimized in [36] only.

b) Lazy evaluation: In order to avoid an overfitting of individuals to a specific data set, appropriate measures must be introduced. Typically (cf. [81]–[83]), different data sets are taken for network training (training data set $\mathcal{L}$) and fitness determination (validation data set $\mathcal{V}$). A cross-validation technique is applied to the training of each individual in each cycle. The alternate optimization of weights and centers with respect to a least-squares error and architecture parameters with respect to a fitness function using different data sets prevents overfitting to a specific data set.

Here, we also apply an n-fold cross-validation technique (n is user-defined), but to keep the runtime of the EA short we spread the cross-validation onto subsequent cycles of the EA. More precisely, the sets of training and validation data are changed in each cycle and the fitness is determined with respect to another cross-validation set (see Fig. 5). Thus, poorly suited individuals can be eliminated quite early, whereas better individuals have to survive several cycles. The idea is that an individual already yielding bad results for one of the data sets will never be among the best anymore. An additional, independent test data set $\mathcal{T}$ must be used to estimate the classification rates that can be achieved in an actual application.

The definition of an individual and the definition of the basis fitness can now be stated more precisely. In the case of an n-fold cross-validation as realized by the lazy evaluation concept each individual of a population consists of:

1) a genotype given by the abstract representation of an individual (see Section IV-B1);
2) altogether n phenotypes for the n training sets given by a complete set of trained network parameters (centers, radii, and weights);
3) a vector $\mathbf{v} \in \mathbb{B}^n$ of additional information with $v_i = 1$ $(i = 1, \ldots, n)$ if the individual has already been trained

on the ith training set (i.e., a valid phenotype exists) and $v_i = 0$ otherwise.

Each valid phenotype is evaluated with respect to another validation set yielding a basis fitness $fit_i^{(basis)}$. The overall basis fitness $fit^{(basis)}$ of an individual is defined by

$$fit^{(basis)} \stackrel{def}{=} \frac{1}{\|\mathbf{v}\|^2} \sum_{i=1}^{n} v_i \cdot fit_i^{(basis)}.$$

This approach gives a chance to recently generated individuals without neglecting older ones.

It is obvious that the speedup factor that can be achieved with lazy evaluation is at most n if compared to a complete n-fold cross-validation in each cycle. The evaluation scheme is "lazy" in the sense that it takes several EA cycles to identify fitter individuals.

Most of the 64 related publications (see Section II) do not apply cross-validation or related techniques such as bootstrapping or hold-out techniques (apart from [54], for instance). Also, different data sets for training, validation, and testing are only used in a few publications (cf. [45], [47], [48], [68], [72], [73], [75], and [80]).

2) Soft Constraints (Side Conditions): It has already been mentioned that the basis fitness $fit^{(basis)}$ is calculated with respect to classification rates or lift factors. Additional soft constraints may be included in the fitness function (cf. Lagrangian approaches for multicriteria optimization):

1) a penalty factor φ_{feature} for the number of features used (to keep the number of free parameters small);
2) a penalty factor φ_{center} for the number of hidden neurons used (for the same reason);
3) a penalty factor φ_{time} for the overall training time of an individual (to control the runtime of the EA);
4) a penalty factor φ_{certain} for the classification safety which is based on the selection function for class members used when calculating the lift factor (to give more importance to networks with clear decisions).

Smaller network structures are easier to interpret, they are less prone to overfit, and their training is faster which leads to a shorter overall runtime of the EA. In some cases, the number of features has to be kept small either because features have to be computed on-line (computational costs) or because features have to be purchased, e.g., from data warehouses (monetary costs). If the training time is punished directly, the runtime is reduced, too. Since smaller network structures can be trained faster, we might expect that shorter runtimes lead to smaller network structures. This, however, has not been shown so far.

The overall fitness of an individual *fit* is calculated by multiplying the basis fitness with penalty factors φ_s for each soft constraint $s \in \{\text{feature}, \text{center}, \text{time}, \text{certain}\}$

$$fit \stackrel{def}{=} fit^{(basis)} \cdot \prod_s \varphi_s.$$

The values of the basis fitness are higher for better individuals. Therefore, the functions used to determine a penalty factor have to be monotonically decreasing with respect to the parameter to be punished. In addition, the relative importance of each penalty factor must be controllable by means of user-defined parameters.

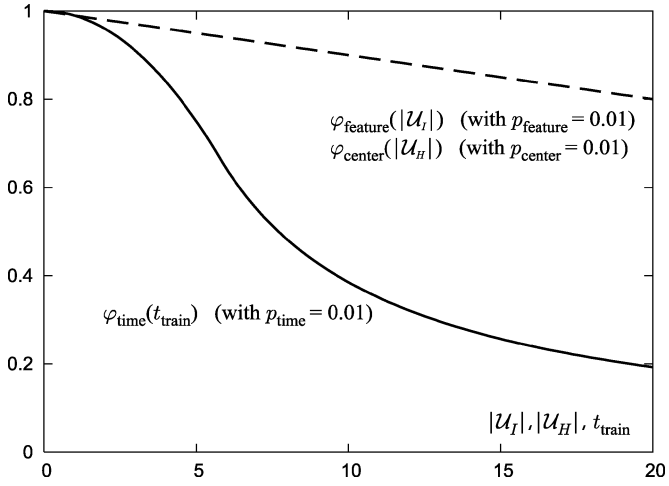

Fig. 6. Examples for penalty factors.

The penalty factor φ_{feature} for the number of features $|\mathcal{U}_I|$ is modeled in the following way:

$$\varphi_{\text{feature}}(|\mathcal{U}_I|) \stackrel{def}{=} 1 - |\mathcal{U}_I| \cdot p_{\text{feature}}$$

where $|\mathcal{U}_I|$ is the number of features in the currently selected feature subset and $p_{\text{feature}} \in \mathbb{R}^+$ is a small, user-defined parameter.

In the same way, the penalty factor φ_{center} that influences the number of hidden neurons $|\mathcal{U}_H|$ is defined

$$\varphi_{\text{center}}(|\mathcal{U}_H|) \stackrel{def}{=} 1 - |\mathcal{U}_H| \cdot p_{\text{center}}$$

where $|\mathcal{U}_H|$ is the number of centers in the individual to be evaluated and $p_{\text{center}} \in \mathbb{R}^+$ is again a small, user-defined parameter. The values of p_{feature} and p_{center} are selected such that φ_{feature}, $\varphi_{\text{center}} > 0$.

The factor φ_{time} for the training time t_{train} is defined by a continuously differentiable and monotonically decreasing function

$$\varphi_{\text{time}}(t_{\text{train}}) \stackrel{def}{=} \begin{cases} 1 - p_{\text{time}} \cdot t_{\text{train}}^2, & \text{for } t_{\text{train}} < \frac{1}{\sqrt{3 \cdot p_{\text{time}}}} \\ \frac{2}{3 \cdot \sqrt{3 \cdot p_{\text{time}}} \cdot t_{\text{train}}}, & \text{otherwise} \end{cases}$$

with a user-defined parameter $p_{\text{time}} \in [0; 1/3)$. This function has the properties $\lim_{t_{\text{train}} \to \infty} \varphi_{\text{time}}(t_{\text{train}}) = 0$, $\varphi_{\text{time}}(0) = 1$, and $\varphi_{\text{time}}(1) = 1 - p_{\text{time}}$. The latter property also holds for the linear models ($\varphi_{\text{feature}}(1) = 1 - p_{\text{feature}}$, $\varphi_{\text{center}}(1) = 1 - p_{\text{center}}$) but larger values of $|\mathcal{U}_I|$ and $|\mathcal{U}_H|$ are punished less dramatically (see Fig. 6).

The factor for classification safety is defined in a similar way [103]. For all factors other penalty models would be applicable, too. In a particular application only a part of the factors is typically used.

The most interesting result of the EA is the *best* individual (network). This is the individual with the highest fitness among all individuals in the last generation of the EA that survived as many cycles as possible (i.e., that have as many valid phenotypes as possible). This network would be used as the best possible classifier in an actual application.

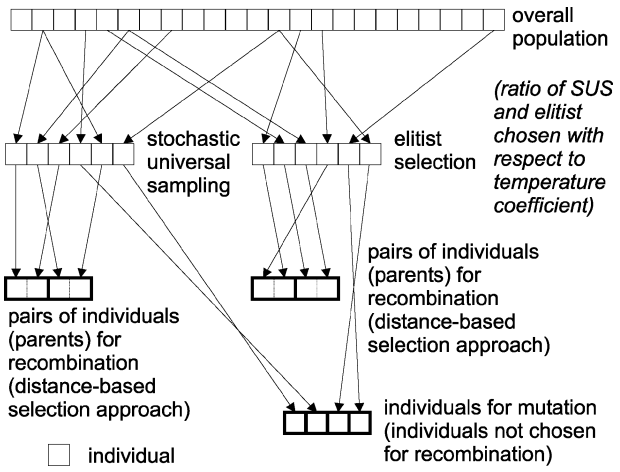

Fig. 7. Selection of parents for reproduction.

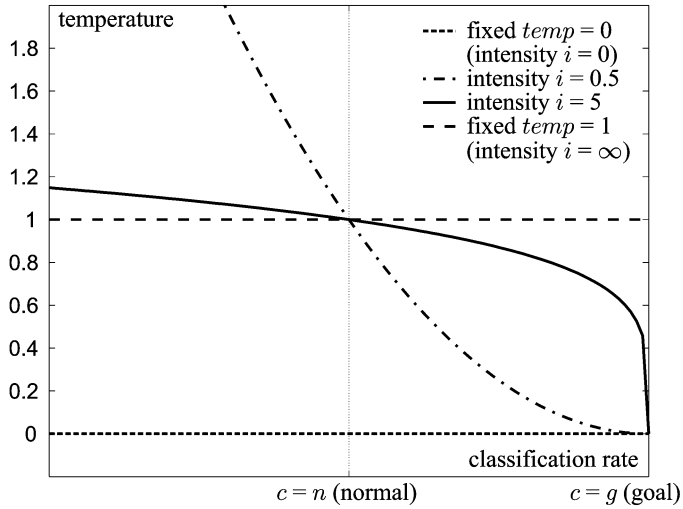

Fig. 8. Examples for temperature coefficient functions.

In some of the 64 related publications (see Section II) penalty terms are used, e.g., for the number of centers (e.g., [30], [31], [36], [56], [58], [65], and [70]), the values of radii (e.g., [30]), the training time (e.g., [36]), the number of weights (e.g., [45] and [54]), and the values of weights (e.g., [38] and [56]). The latter approach is quite similar to *weight decay* in neural network training [11]. In most publications the influence of penalty terms can not be weighted, and in none of them this idea is investigated in detail. Closely related are multicriteria optimization approaches where a set of Pareto optimal solutions (nondominated solutions) is sought for. Examples can be found in [18], [40], [48], [52], [54], [62], [70], and [77]. Typical criteria are mean and maximum classification errors, number of centers, and number of weights.

3) Temperature Coefficient: The use of a temperature coefficient allows an adaptive control of the evolutionary optimization process (coarse tuning versus fine tuning). It depends on the classification rates of the fittest individual in the current population or, alternatively, its lift factor. The coefficient is high for low classification rates and vice versa. A high value of this coefficient allows large steps in the search space in order to explore new regions. If the value is low, a local search around the current position in the search space is supported. Thus, a suitable compromise between fast search (low runtime) and exhaustive search (low classification error) may be effected. Premature convergence in local minima can be avoided without sacrificing runtime efficiency.

The temperature coefficient has a significant influence on the values of stochastic parameters used by selection and mutation operators.

Fig. 7 describes the utilization of the temperature coefficient for the selection of parents for reproduction. With a high temperature coefficient, SUS selects a larger set of parents in order to give a chance to less optimal individuals.

Here, mutation must accomplish two goals whose importance is controlled by the temperature coefficient. First, completely new regions in the search space should be reached. Second, a local search should be conducted when an optimum is approached. In this respect, the temperature coefficient has an impact on the probability used for the mutation of the training method and on the standard deviations of the random variables needed to mutate the number of features selected, the number of centers, and the training time.

To implement these ideas, the following function for temperature calculation was chosen:

$$temp(c) \stackrel{def}{=} \begin{cases} 0, & \text{for } c > g \\ 0, & \text{for } i = 0 \\ \left(\frac{g-c}{g-n}\right)^{1/i}, & \text{otherwise} \end{cases}$$

with c being the classification rate of the currently best individual, i a positive intensity value, g the goal classification rate, and n (with $n \leq g$) the classification rate for which $temp(n) = 1.0$ ("normal" temperature). With the intensity parameter $i \in [0, \infty)$ the temperature function varies smoothly between $temp \equiv 0$ and $temp \equiv 1$. In the remainder, the notation $i = \infty$ is used for $\lim_{i \to \infty} temp(c) = 1$, i.e., a fixed temperature $temp = 1$.

Fig. 8 shows the temperature curves for intensities $i \in \{0, 0.5, 5, \infty\}$. A low intensity causes the temperature to decrease far from the goal classification rate. A high intensity implies that the temperature stays at a high level until the goal classification rate is nearly reached.

According to the value of the temperature coefficient the EA must perform a finer or coarser search. One way to achieve this behavior is varying between SUS and elitist selection for mutation and recombination in the following way:

$$s_{\text{elitist}}(temp) \stackrel{def}{=} \begin{cases} 0, & \text{for } temp > 1.0 \\ (1 - temp) \cdot s, & \text{otherwise} \end{cases}$$

where s is the number of individuals to be selected for mutation or for recombination, respectively. The resulting values are rounded to the nearest integer value. Consequently, the number of individuals selected by SUS is

$$s_{\text{sus}}(temp) \stackrel{def}{=} s - s_{\text{elitist}}(temp).$$

The mutation of centers is influenced by the temperature in the following way:

$$hidden_{\text{new}} = hidden + \mathcal{N}(0, temp \cdot mut_{\text{hidden\_std}})$$

where *hidden* is the number of hidden neurons (centers) of the individual to be mutated and $\mathcal{N}(0, temp \cdot mut_{\text{hidden\_std}})$ is the realization of a normally distributed random variable with expectation 0 and standard deviation $temp \cdot mut_{\text{hidden\_std}}$. The value $mut_{\text{hidden\_stdd}}$ is a user-defined standard deviation for mutation at normal temperature. Again, the resulting values are rounded to the nearest integer.

Almost all of the 64 related publications (see Section II) do not utilize an adaptive control of the evolution process. Exceptions are [34] and [35] which employ a dynamic mutation rate that depends on observations indicating a stagnation of the evolution process.

V. Experimental Results

In this section, we demonstrate the properties and advantages of our approach by means of four real-world application examples:

- **ID**: intrusion detection (network-based misuse identification) in computer networks;
- **SV**: signature verification (biometric, behavioral-based authentication);
- **DM**: direct marketing campaign (customer relationship management);
- **PO**: chemical production optimization (by identification of optimal working points).

For our purposes here, all four application examples are interesting for various reasons: The former two (ID and SV) came from research projects, the latter two (DM and PO) arose in industrial projects. Three examples (ID, SV, and DM) have already been investigated intensely applying other techniques and/or earlier versions of our EA-based approach which makes comparisons possible. The examples possess different numbers of possible features (up to 137), classes (up to 4), and patterns (up to approximately 80 000). The optimization criteria are different: Lift factor in the case of DM, classification rates otherwise. The auxiliary objectives vary, too: Small number of features, interpretable network structures, or short time for modeling. The models obtained by means of the EA would later be applied either for on-line data processing (ID and SV) or off-line data analysis (DM and PO). The data have classes with very different separability properties. Finally, the innovative extensions described in Section IV-C cannot be satisfactorily validated by only one single data set. Apart from the investigations shown here we also carried out experiments with generic data and data taken from the Proben1 or the UCI benchmark collections [104], [105].

The final outcome of the EA is the best network as described in Section IV-C2. Therefore, the classification error rates of this network (or lift factors in the case of DM) on validation data used by the EA to determine the individual's fitness and on independent test data not taken for any optimization step ($\mathcal{E}$ vali, $\mathcal{E}$ test, lift vali, lift test) are investigated. $\mathcal{E}$ test and lift test estimate the individual's behavior in an actual application. Additionally, the number of features, the number of centers (i.e., basis functions), and (in some cases) the training time are given for the best networks.

Another interesting result is the behavior of the EA for a given parameterization and data set. Particularly, its runtime, the overall number of individuals evaluated, the number of different combinations of input features investigated, and the number of different center numbers tested are shown here.

TABLE I
Most Important Parameters of the EA

Application Example	ID	SV	DM	PO
Data Set				
number of features	137	93	57	24
overall number of samples	1 914 – 78 408	600	18 942	1 092
number of classes	2	2	2	4
Population				
population size	40	40	15	40
# individuals selected for mutation	15	15	5	15
# pairs of individuals selected for recombination	10	10	4	10
Network Evaluation				
basis fitness: lift factor (l) or classification error (c)	c	c	l	c
# validation sets (cross-validation)	5	10	5	10
size of $\mathcal{L}$ (training data)	1 016 – 52 800	504	3 970	738
size of $\mathcal{V}$ (validation data)	254 – 13 200	56	8 323	82
size of $\mathcal{T}$ (test data)	644 – 12 408	40	6 649	272
penalty values (if used)				
p_{feature}	0.0001	0.00001	0.01	0.035
p_{center}	0.0001	0.001	0.005	0.0001
p_{time}			0.02	
p_{certain}			5	
EA Control				
# of cycles	50	200	100	100

The best network obtained by the EA is only useful if the results of the EA are repeatable. Here, the reliability of the EA is measured by repeatedly starting the EA with different, randomly chosen initial populations. Each experiment consists of 50 runs (repetitions) and average values (μ) as well as empirical standard deviations (σ) of the criteria mentioned above are shown. Although the EA has more than 40 parameters, only a few of them have a relevant impact on the best network. Examples for less important parameters are initial values of $|\mathcal{U}_I|$, $|\mathcal{U}_H|$, and t_{train} or lower and upper bounds of the training time. The most important parameters are summarized in Table I together with the values utilized in Section V-A.

The experiments have been conducted on 2.8 GHz Intel Pentium IV computers with Linux (Red Hat) operating system and 1-GByte memory.

The remainder of this section is structured as follows: First, the best results that could be achieved for each of the four application examples are described. Next, the benefits of the various innovative extensions are investigated. Finally, the new approach is compared to our own, earlier EA-based approach that has already been published (ID and DM).

A. Application Problems

1) Intrusion Detection—ID: With the rapidly increasing impact of the Internet there is also an emerging need for data and information security. Various techniques such as authentication (see the following subsection), data encryption, firewalls, or *intrusion detection systems* (*IDS*) help to protect against attacks (cf. [106] and [107]). For more than a decade, the utilization

TABLE II
INTRUSION DETECTION—OVERALL RESULTS

Experiment No.	ID-N-1		ID-P-1		ID-S-1	
Attack Type	Nmap		Portsweep		Satan	
Best Networks						
	μ	σ	μ	σ	μ	σ
$\mathcal{E}$ vali in %	0.00	0.01	0.54	0.37	0.47	0.03
FA vali in %	0.00	0.02	0.69	0.67	0.36	0.04
MA vali in %	0.00	0.00	0.36	0.38	0.60	0.07
$\mathcal{E}$ test in %	0.00	0.02	0.34	0.19	0.57	0.04
FA test in %	0.00	0.04	0.38	0.16	0.34	0.04
MA test in %	0.00	0.00	0.29	0.39	0.81	0.09
# centers	1.16	0.37	9.66	6.33	4.60	3.73
# features	1.40	0.61	7.22	2.98	3.28	1.62
Behavior of the EA						
	μ	σ	μ	σ	μ	σ
runtime in min	1.17	0.34	34.02	16.33	151.12	51.68

of soft computing techniques has been investigated for different kinds of intrusion detection such as misuse detection or anomaly detection.

The most important evaluations of IDS performed up to now were supported by the Defense Advanced Research Projects Agency (DARPA) in 1998 and 1999 [108], [109]. Here, the data of the first DARPA IDS evaluation is used. RBF networks are applied to classify a communication as being either a certain attack or normal behavior (the latter includes other attack types). The following three attack types are considered: Nmap, Portsweep, and Satan. A set of 137 possible features (the same for all attacks) is extracted from the raw tcpdump output provided by the DARPA. Only protocol information, i.e., TCP and IP header information, and not the transmitted data are used. Examples of features actually selected by the EA are the average number of packets in a connection containing the FIN (finish) flag (Nmap), the average number of packets in a connection containing the RST (reset) flag (Portsweep), and the number of packets sent by the client containing the FIN flag (Satan).

At a more abstract level, the application problem can be specified as follows: We are given a set $\mathcal{F}$ of possible features ($|\mathcal{F}| = 137$) and for each attack type a a set $\mathcal{K}_a = \{(\mathbf{m}, c)\}$ of feature vectors $\mathbf{m} \in \mathbb{R}^{|\mathcal{F}|}$ with class labels $c \in \mathbb{B}$ ($1\,914 \leq |\mathcal{K}_a| \leq 78\,408$ depending on the attack type). Each $\mathcal{K}_a$ describes two classes: Attacks of a certain type that must be detected (a) and normal communication behavior or other attacks. Each $\mathcal{K}_a$ must be divided up into training, validation, and test sets. The objective is to find attack-specific subsets $\mathcal{F}'_a \subseteq \mathcal{F}$ of features and attack-specific mappings $f_a : \mathbb{R}^{|\mathcal{F}'_a|} \rightarrow \mathbb{B}$ that determine the class membership for any input vector $\mathbf{m'}_a \in \mathbb{R}^{|\mathcal{F}'_a|}$ with respect to a certain attack. Additionally, the feature subset should be small (i.e., $|\mathcal{F}'_a| \ll |\mathcal{F}|$) because the features have to be computed on-line. For the same reason, the evaluation of f_a should be fast. Also, f_a should be interpretable, i.e., the network structure should be small, so that comprehensible rules could be extracted.

Results are shown in Table II. As mentioned above, μ and σ stand for the average values and the empirical standard deviations of various criteria. The classification error for unknown test patterns ($\mathcal{E}$ test in %) is very low for these attack types, which indicates a good generalization ability. Table II also shows the rates of false alarms (FA) and missing alarms (MA). The number of features is very low for all attack types.

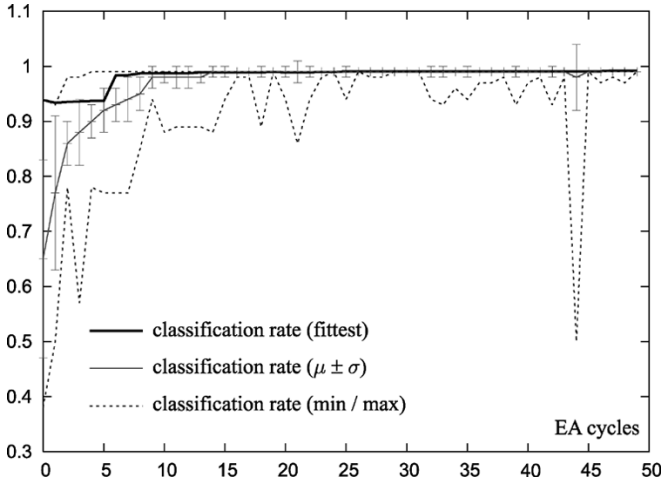

Fig. 9. Example for the development of classification rates (validation data) in a population (portsweep detection).

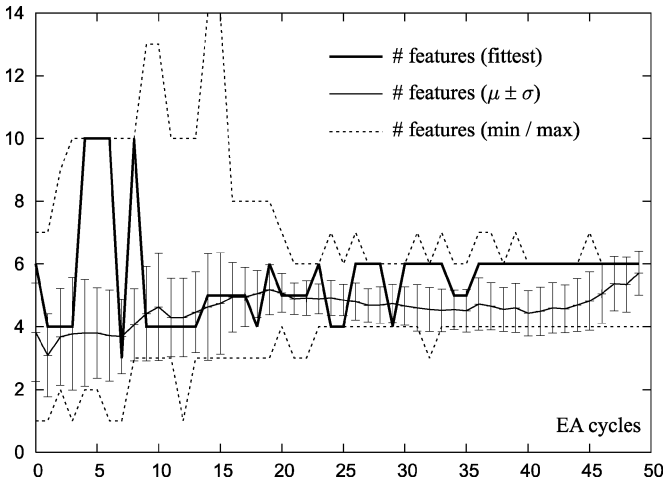

Fig. 10. Example for the development of the feature number in a population (portsweep detection).

Only between one and seven of the 137 features are actually needed to detect an attack. The feature subset selected by the EA is different for each attack type. This outlines the need for attack-specific modeling. The number of centers is quite low, too. It is very important from a practical point of view that these results can be achieved with the same EA parameterization for each attack type. The runtime of the EA strongly depends on the size of the subset of $\mathcal{K}_a$ used for training. An analysis of the development of classification rates and number of features in the population shows that shorter runtimes are possible. Typically (see Figs. 9 and 10), the EA converges after about 10–20 cycles for this application and parameter setting.

The DARPA data set has been used by many researchers up to now (see [110]–[112], for instance). It was also the basis for the KDD Cup 1999 that has been won by an approach based on decision trees with a combination of bagging and boosting (overall classification error 7.67%) [113]. A direct comparison, however, would not be fair because many additional attack types had to be detected etc. Our own work with this data set includes a comparison of various neural and fuzzy classifier paradigms including an ensemble approach [114], an earlier evolutionary approach without the innovative extensions introduced here [81], and the

extraction of understandable rules for intrusion detection from trained neural networks [115].

2) Signature Verification—SV: Biometric authentication methods belong to another, very important category of techniques that ensure data and information security. Here, an example in the field of signature verification is investigated (cf. [116] and [117]). A person provides her/his signature together with a personal identification number (PIN) and a verification system must decide whether the signature corresponds to this PIN or not. Possible application areas are credit card payment processes, access control systems (in airports, for instance), or inspections by police or customs.

The verification approach proposed here is based on a biometric pen that provides three force and acceleration signals in orthogonal directions [118], [119]. In the following, results are shown for an approach that computes 93 different features for each signature. Typical examples are the overall length of the signature, the pen-up time, the number of strokes, minimum and maximum values, various parameters of stochastic models such as skewness and kurtosis, etc. Individual RBF networks for each person classify a signature as being either genuine or forged (i.e., the signature of another person). Here, models are set up for three persons and tested against about 75 other persons.

Again, at a more abstract level, the problem can be stated as follows: We have a set $\mathcal{F}$ of possible features ($|\mathcal{F}| = 93$) and for each person p a set $\mathcal{K}_p = \{(\mathbf{m}, c)\}$ of feature vectors $\mathbf{m} \in \mathbb{R}^{|\mathcal{F}|}$ with class labels $c \in \mathbb{B}$ ($|\mathcal{K}_p| = 600$). Each $\mathcal{K}_p$—that must be divided up into training, validation, and test sets—describes two classes: Signatures of a certain person (p) and other signatures. We are seeking for person-specific subsets $\mathcal{F}'_p \subseteq \mathcal{F}$ of features and person-specific mappings $f_p : \mathbb{R}^{|\mathcal{F}'_p|} \rightarrow \mathbb{B}$ that determine the class membership for any input vector $\mathbf{m'}_p \in \mathbb{R}^{|\mathcal{F}'_p|}$. Again, the feature subset should be small (i.e., $|\mathcal{F}'_p| \ll |\mathcal{F}|$) because the features have to be computed on-line. For the same reason, the evaluation of f_p should be fast. Additionally, the time required for model building should be short because in an actual application models need to be built for a large number of persons.

Table III shows experimental results. The error rates for unknown test data are quite low (FAR and FRR are the false acceptance rate and the false rejection rate.). With improved pre-processing techniques (signature segmentation, feature extraction, etc.) we expect to improve these results further. Our main discoveries are the facts that the runtime of the EA is relatively low for each person, good results can be achieved with the same EA parameterization for each person, and the number of features actually needed is relatively low. Table IV outlines the claim for person-specific modeling (in particular: feature selection). It sets out the features that are used by at least 75% of the best networks in the 50 runs. There are a few features that are selected for all persons, such as the length of the signature (N01) or the pen-up time (N03). There are also some other features that are needed to distinguish a particular person from other persons (e.g., spectral features such as RA49).

The same pen has been used in [118] and [120]–[122] for signature verification but with other data sets and features. Verification for a group of 40 writers applying conventional (hard computing) classification algorithms yielded error rates between 10% and 23% for data collected at subsequent days

TABLE III
SIGNATURE VERIFICATION—OVERALL RESULTS

Experiment No.	SV-Bu-1		SV-Gr-1		SV-Gü-1	
Name	Buchtala		Gruber		Gürster	
Best Networks						
	μ	σ	μ	σ	μ	σ
$\mathcal{E}$ vali in %	3.93	0.37	1.24	0.25	1.46	0.47
FRR vali in %	2.75	0.96	0.61	0.60	1.14	0.75
FAR vali in %	5.12	0.86	1.88	0.39	1.79	1.19
$\mathcal{E}$ test in %	4.90	0.71	3.00	1.82	1.35	1.97
FRR test in %	1.70	2.39	0.00	0.00	0.00	0.00
FAR test in %	8.10	2.45	6.00	3.64	2.70	3.94
# centers	3.92	2.78	3.76	0.77	3.30	2.05
# features	23.78	5.14	32.66	7.03	30.42	6.71
Behavior of the EA						
	μ	σ	μ	σ	μ	σ
runtime in min	8.86	2.86	10.06	1.49	10.16	1.62

TABLE IV
IMPORTANCE OF FEATURES—SIGNATURE VERIFICATION.

Feat. No.	Buchtala	Gruber	Gürster
Before Resampling and Normalization			
N01	98%	100%	98%
N02	76%		
N03	98%	100%	94%
After Resampling and Normalization			
RA01		92%	
RA04		100%	
RA05		100%	
RA13		90%	
RA38			100%
RA49		76%	
RA52	78%		
RA54			84%
RA58			98%
RA59			96%
RA60			82%
RA70	76%		
RA71			80%

[118]. With Adaptive Resonance Theory Networks (ART-2) and a group of ten writers error rates between 5.7% and 17.7% were achieved in [121] and between 0.0% and 14.3% in [120] (for the best parameter set). The best and most recent results are described in [122], where false acceptance rates and false rejection rates between 4.5% and 9.5% are mentioned for ART-2 and self-organizing maps (SOMs). Again, the experiments were conducted with a group of ten writers. Here, models for three writers were tested against about 75 other writers. For the best parameter settings in a dynamic approach (time series classification) we achieved error rates on test data between 1.15% and 4.56% only [123].

3) Customer Relationship Management—DM: The classical data mining application investigated here arose in the Sales Department of DaimlerChrysler AG, when a direct mailing campaign targeting the launch of the new Mercedes Benz E-Class was planned (cf. [124] and [125]). Promising addressees had to be selected on the basis of so-called *micro-geographical* data, i.e., aggregated information on small geographical units (micro-cells) such as the size of the city, the status of the residents with respect to education and income, or the fluctuation in the micro-geographical unit.

The objective is to set up a model (RBF network) that relates micro-geographical features describing a micro-cell to an output variable indicating whether a person receiving a direct mail will

TABLE V
DIRECT MARKETING—OVERALL RESULTS

Experiment No.	DM-1	
Best Networks		
	μ	σ
lift vali	2.53	0.10
lift test	2.21	0.27
# centers	1.00	0.00
# features	6.48	1.87
training time in s	0.04	0.03
Behavior of the EA		
	μ	σ
runtime in min	8.86	2.49

TABLE VI
IMPORTANCE OF FEATURES—DIRECT MARKETING

Rate	Description
100%	Proportion of cars of other brands (not: Audi, BMW, Ford, Fiat, Honda, Mercedes, Nissan, Opel, Peugeot, Renault, Toyota, VW)
56%	Unit belongs to social group with percentage of buyers above or below average (binary)
46%	Percentage of buyers in social group
38%	Status (education and income) above or below average (binary)
38%	Relocations within the district unit
36%	Size of the city

buy a Mercedes Benz automobile. Binary output data ("buyer" or "nonbuyer") gained from past promotions is available (18 942 samples with 57 features). The trained network would finally be used to assess all German micro-cells. This evaluation, called *scoring*, regards only the micro-cells in the top percentile of the scores (defined by the so-called *cut-off point*) as promising with respect to the target variable. This problem can be seen as a typical sample subset selection problem (see Section IV-B3).

Again, we describe the task in a slightly more formal way: We are given a set $\mathcal{F}$ of possible features ($|\mathcal{F}| = 57$) and a set $\mathcal{K} = \{(\mathbf{m}, c)\}$ of feature vectors $\mathbf{m} \in \mathbb{R}^{|\mathcal{F}|}$ with class labels $c \in \mathbb{B}$ ($|\mathcal{K}| = 18\,942$). $\mathcal{K}$ (that must be divided up for training, validation, and testing) describes purchase decisions in prior campaigns. The goal is to find a subset $\mathcal{F}' \subseteq \mathcal{F}$ of features and a mapping $f : \mathbb{R}^{|\mathcal{F}'|} \to [0, 1] \subset \mathbb{R}$ that estimates the purchase probability for an input vector describing a micro-cell. The mapping f must be applied to a set $\mathcal{P}$ (with $|\mathcal{P}| \approx 30\,000\,000$) containing feature vectors of all German micro-cells and a subset $\mathcal{P}' \subset \mathcal{P}$ with $|\mathcal{P}'| = 0.05 \cdot |\mathcal{P}|$ must be chosen for which the purchase probability is highest. Again, the feature subset should be small (i.e., $|\mathcal{F}'| \ll |\mathcal{F}|$) because the purchase of the scoring data is expensive. Also, the evaluation of f should be fast, because f must be evaluated for some millions of input vectors.

The results given in Table V show that lift factors (cf. the definition in Section IV-B3) of about 2.21 can be achieved for unknown test data. The feature number needed for this result is low. Table VI sets out the features that are selected in at least 33% of the best networks. The first feature has been selected in every run; the second and the third are highly correlated (i.e., exchangeable). Fig. 11 gives a typical example for the development of the lift factor in the population. The best networks with only one hidden neuron can be evaluated very fast. As a consequence, the scoring of some millions of input vectors could be conducted within a few hours.

This marketing problem has been investigated in detail in [126]. There, an evolutionary approach for feature selection was tested as well as filter techniques using a class separability measure. The best solution achieved by means of the EA had a lift factor of 3.45 with respect to validation data, which outperformed all previously found solutions by far. However, testing this solution on another, independent test set revealed that it has been significantly overfitted (lift factor of 1.95). Other approaches that have been applied to the same data set utilized multilayer perceptrons or decision trees. With these techniques, lift factors of about 1.5–1.8 could be achieved.

4) Optimization of Chemical Production Processes—PO: Starting point for the production of integrated

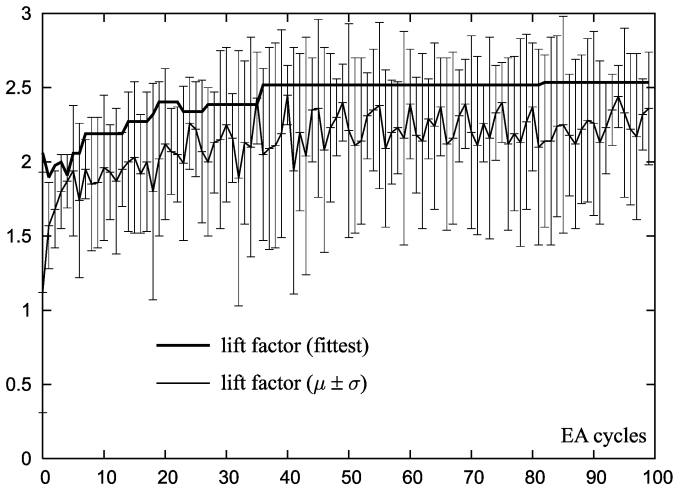

Fig. 11. Example for the development of lift factors (validation data) in a population.

circuits are wafers, i.e., thin slices of single crystals of silicon. The wafer production process (see [127] for some more details) starts with sand that is reduced to elemental silicon in an electrothermal reaction. This crude silicon contains roughly one percent of impurities. The next step in purification of this silicon is a reaction between silicon and hydrogen chloride to yield trichlorosilane and hydrogen

$$\mathrm{Si} + 3\mathrm{HCl} \longrightarrow \mathrm{SiHCl_3} + \mathrm{H_2}.$$

By distillation at a relatively low temperature the impurity level is lowered to less than 1 ppb (part per billion), and the pure trichlorosilane is converted back into elemental, polycrystalline silicon. Then, single crystals of silicon are produced from the melted polycrystalline silicon.

Here, the focus is on improvements of the chemical reaction mentioned above. By-products such as silicon tetrachloride ($\mathrm{SiHCl_4}$) are generated during this process as well. *Selectivity* is a measure that describes the proportion of $\mathrm{SiHCl_3}$. It is usually desired to be as high as possible, and it depends on three types of process variables: The composition of the input to the process (feed), the amount of chemical impurities in the reactor, and the process conditions. In order to increase selectivity, the most important of these parameters must be identified. An optimal process behavior can then be achieved by varying the significant variables.

The process identification problem is modeled as a classification problem with four classes describing the degree of selectivity (low, medium, high, and very high) with respect to

TABLE VII
PROCESS OPTIMIZATION—OVERALL RESULTS

Experiment No.	PO-1	
Best Networks		
	μ	σ
$\mathcal{E}$ vali in %	5.06	1.99
$\mathcal{E}$ test in %	6.50	2.15
# centers	136.89	13.17
# features	3.00	0.85
Behavior of the EA		
	μ	σ
runtime in min	51.31	6.99

TABLE VIII
PROCESS OPTIMIZATION—CLASSIFICATION RATES ON TEST DATA

		Predicted Class			
		low	medium	high	very high
Correct	low	**23.30%**	1.18%	0.34%	0.18%
Class	medium	0.35%	**23.35%**	1.18%	0.12%
	high	0.09%	1.80%	**22.73%**	0.38%
	very high	0.11%	0.15%	0.57%	**24.17%**

the process variables. This degree of selectivity is predicted by an RBF network. The best network found by the EA is subsequently analyzed to find the optimal working point of the reactor. The investigations described here were conducted together with Wacker-Chemie GmbH and Siltronic AG.

Again, we describe the task in a more formal way: We start with a set $\mathcal{F}$ of possible features ($|\mathcal{F}| = 24$) and a set $\mathcal{K} = \{(\mathbf{m}, c)\}$ of feature vectors $\mathbf{m} \in \mathbb{R}^{|\mathcal{F}|}$ with a class label $c \in \{\text{low}, \text{medium}, \text{high}, \text{very high}\}$ ($|\mathcal{K}| = 1\,092$). $\mathcal{K}$ describes the dependencies between process variables and selectivity classes and must be used for training, validation, and testing. We search for a subset $\mathcal{F}' \subseteq \mathcal{F}$ of features and a mapping $f : \mathbb{R}^{|\mathcal{F}'|} \rightarrow \{\text{low}, \text{medium}, \text{high}, \text{very high}\}$ that predicts the selectivity for a certain process variable setting.

Results for this data set are set out in Table VII. The degree of selectivity can be predicted with a surprisingly low error rate, and it can be controlled by only three of the 24 process variables. Table VIII shows the accuracy of the best networks in a scatter matrix that sets out the error types in some more detail.

B. Specific Questions

Now, the innovative aspects of our approach will be investigated in greater detail:

- the fast fitness evaluation of individuals;
- the use of soft constraints (side conditions);
- the temperature-based control of the EA.

The data sets described above will be used again, but parameters are not necessarily chosen such that classification rates or lift factors are optimal. Instead, values of parameters are selected such that the effects of our extensions become obvious.

1) Fast Fitness Evaluation: The main objective of hybrid RBF training and lazy evaluation of individuals is a substantial reduction in runtime.

a) Hybrid RBF Training: The advantages of the hybrid training concept that combines modified k-means, QR decomposition (QRD), and SCG (Section IV-C1a) have already been investigated in detail in [85], [86] but with Backpropagation (BP) and Resilient Propagation (RPROP, [128]) instead of SCG. The conclusions drawn there can be transferred to the slightly modified training concept applied here. Therefore, only the main findings will be summarized.

TABLE IX
LAZY EVALUATION—INTRUSION DETECTION

Experiment No.	ID-P-2		ID-P-3	
Best Networks				
	μ	σ	μ	σ
$\mathcal{E}$ vali in %	0.60	0.39	0.60	0.42
$\mathcal{E}$ test in %	0.42	0.24	0.41	0.23
# centers	5.96	4.04	7.12	5.24
# features	6.80	3.14	5.74	2.66
Behavior of the EA				
	μ	σ	μ	σ
runtime in min	23.81	9.77	45.75	17.68
# trained netw.	2 722.44	163.60	6 450.00	0.00
# feature comb.	255.74	83.29	221.26	71.72
# center numb.	21.46	4.61	21.32	5.18

In general, it can be expected that any iterative weight adaptation algorithm (BP, RPROP, SCG) may yield lower training errors than the combination of k-means and QRD alone. Conversely, the latter may be significantly faster than the former. The question is whether the combination of k-means, QRD, and SCG (or BP/RPROP) effects a suitable compromise between accuracy and speed. For the DM data set it is shown in [86] that the combination of k-means, QRD, and BP yields about 29% lower training errors than k-means + QRD and—to achieve the same error— it is about 42% faster than BP alone. With an ID data set similar results are given in [85]: the combination of k-means, QRD, and RPROP yields about 79% lower training errors than k-means + QRD and—with a given error threshold—it is about 28% faster than RPROP alone. In general, an RBF network should be trained in two subsequent steps. The first step (k-means + QRD) reduces the error significantly and with minimal temporal effort. The second step (iterative weight adaptation with BP, RPROP, or SCG) usually lowers the error further but at relatively high temporal expenses. The length of this step should, therefore, be adapted by the EA.

b) Lazy Evaluation: The following experiment shows that lazy evaluation of individuals yields a significant reduction in runtime. Compared to conventional cross-validation the same classification rates can be achieved and overfitting does nor occur. Table IX sets out the results for an ID data set with 5-fold cross validation (ID-P-2 with lazy evaluation, ID-P-3 with standard cross-validation). It can be noticed that the runtime is reduced by about 48%. The number of trained networks is smaller, too. Without lazy evaluation the number of trained networks is the same in each run of the EA. Similar observations could be made for the other data sets. In general, a lot more different solutions (in particular due to different feature combinations) are investigated within a significantly shorter amount of time.

2) Soft Constraints: In the following, the influence of soft constraints (side conditions) on the results of the EA is investigated. It is shown that there are data sets and applications where these constraints significantly improve results. It should also be mentioned that there are interdependencies between certain constraints (center number versus training time and feature number versus training time).

a) Feature number soft constraint: The primary objectives of a penalty on the number of features are to reduce costs

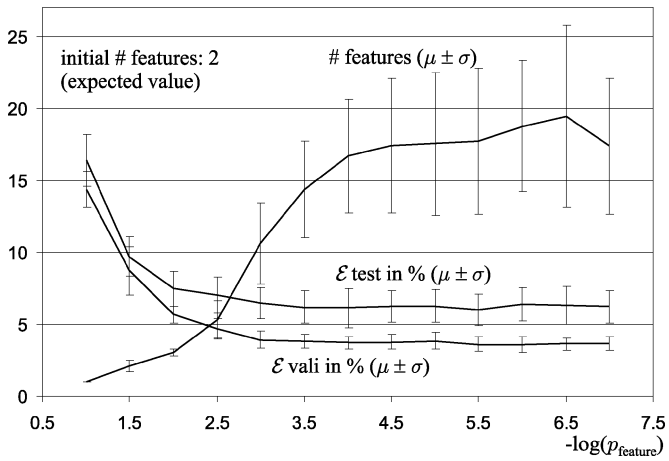

Fig. 12. Feature number soft constraint—signature verification.

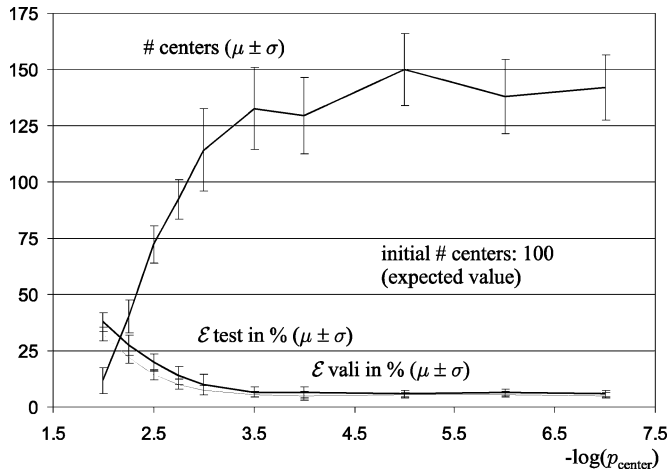

Fig. 13. Center number soft constraint—process optimization.

(computational or monetary) and to obtain interpretable networks. It can be expected that with a severe penalty the number of features decreases whereas the error rates on validation and test data increase. Fig. 12 shows this behavior for an SV data set. Note that the abscissa shows $-\log(p_{\text{feature}})$. That is, the degree of the penalty falls from the left to the right.

It is quite easy to effect a compromise between a low error rate and a low feature number (and a short runtime, which is highly correlated). It can also be stated that even with a large number of features the EA does not overfit with respect to validation data (cf. Section IV-C1b). With a severe penalty on the number of features, the number of centers decreases slightly (not shown).

Experiments on other data sets confirm that the effects of an increasing penalty on the number of features can first be observed when the penalty value p_{feature} approaches about a tenth of the error rate.

b) Center number soft constraint: The main objectives of a penalty on the number of centers are comparable to those of a penalty on the number of features. It can be expected that with respect to values of the parameter p_{center} a similar behavior of the error rates can be observed. Fig. 13 shows this behavior for the PO data set. A trade-off between low error rate and low number of centers can easily be attained. The number of features is approximately the same for all values of p_{center}.

c) Training time soft constraint: Primary objectives of a penalty on the training time are to reduce the overall runtime of the EA and to make the network structures smaller (indirect penalty of feature and center numbers, cf. [63]). As a consequence, more solutions can be investigated with the same temporal effort. It can be expected that with a severe penalty the error rates on validation and test data increase but it is not clear how the number of features and the number of centers are affected.

Fig. 14 shows results of the EA on an ID data set with various values of p_{time}. It can be noticed that the training time of the best networks (the parameter which is optimized by the EA) is reduced significantly with a severe penalty. As a consequence, the overall runtime of the EA is reduced, too (from an average time of 41.34 h to 7.65 h). The error rates increase significantly but there is definitely no clear trend concerning the number of features and the number of centers. It can be concluded that a penalty on training time is not an appropriate measure when the

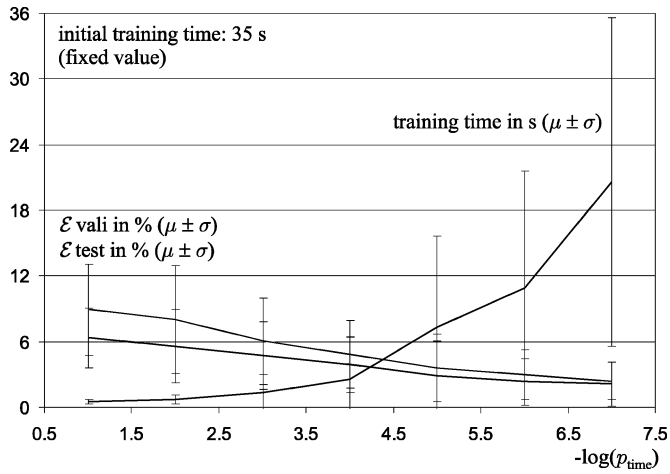

Fig. 14. Training time soft constraint—intrusion detection.

network structures should be kept small. In this case, a direct penalty on the numbers of centers and/or features should be considered.

d) Classification certainty soft constraint: The evaluation of the classification certainty constraint is accomplished with the DM data set. With a high value of the parameter p_{certain} we expect the EA to prefer networks that have both, a high lift factor which is closely correlated to the classification rate and a high value of the safety term which is closely correlated to the MSE of the network (see Section IV-C2). It turns out that it is difficult to chose appropriate values for the parameter p_{certain}. One reason is certainly the fact that the classification error rate and the MSE of a network are not strongly correlated.

Table X shows the results of two experiments: In experiment DM-1 (already discussed above), the constraint is used with $p_{\text{certain}} = 5$; in experiment DM-2, the constraint is switched off. It can be noticed that the lift factors for validation and test data are significantly higher with the utilization of the constraint.

3) Temperature Coefficient: The behavior of the temperature-based control of the EA is illustrated using an SV data set. The intensities $i \in \{0, 0.5, 1, 2, 5, \infty\}$ are tested in these experiments. Fig. 15 shows the development of the temperature in four experiments with intensities $i \in \{0.5, 1, 2, 5\}$.

TABLE X
CLASSIFICATION CERTAINTY SOFT CONSTRAINT—DIRECT MARKETING

Experiment No.	DM-1		DM-2	
Best Networks				
	μ	σ	μ	σ
lift vali	2.53	0.10	2.41	0.15
lift test	2.21	0.27	1.92	0.37
# centers	1.00	0.00	1.28	0.70
# features	6.48	1.87	5.58	2.23

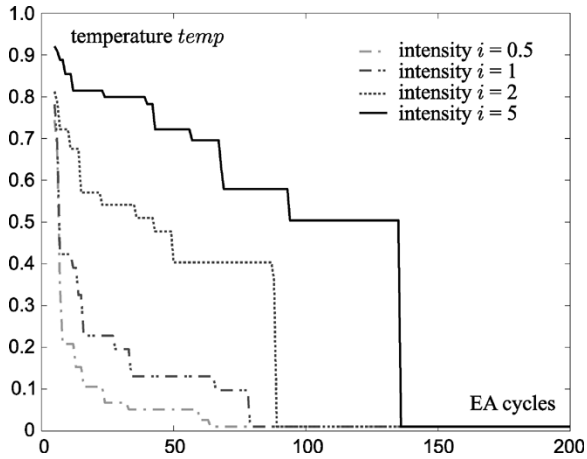

Fig. 15. Examples for the development of temperature values—signature verification.

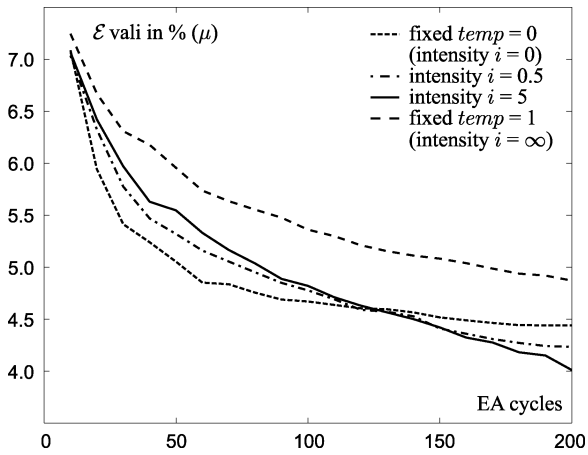

Fig. 16. Classification error rates (validation data) in temperature experiments—signature verification.

It is expected that a low intensity increases the chance of running into local minima. On the other hand, a high intensity reduces the efficiency of the evolutionary search, i.e., it takes more cycles to achieve good classification rates. Fig. 16 sets out the errors on validation data (plotted for intensities $i \in \{0, 0.5, 5, \infty\}$). The curve for $i = 0$ decreases quickest at the beginning but converges after about 60 epochs (local minima). On the other hand, the curve for $i = \infty$ ($temp = 1$) descends slowly. A good compromise which outperforms all other experiments after about 130 epochs is attained with $i = 5$. Fig. 17 shows the errors for test data. Intensity $i = 0$ yields higher, $i = \infty$ lower test errors. Again, the best results can be achieved with $i = 5$.

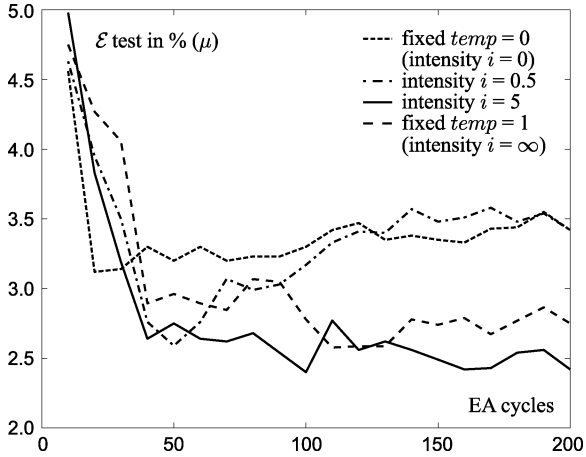

Fig. 17. Classification error rates (test data) in temperature experiments—signature verification.

TABLE XI
OVERALL IMPROVEMENTS—DIRECT MARKETING

Experiment No.	DM-1		DM-3	
Best Networks				
	μ	σ	μ	σ
lift vali	2.53	0.10	2.27	0.13
lift test	2.21	0.27	1.80	0.30
# centers	1.00	0.00	5.40	5.97
# features	6.48	1.87	12.62	4.73
Behavior of the EA				
	μ	σ	μ	σ
runtime in h	0.15	0.04	3.20	0.15
# trained netw.	1 681.22	63.93	3 050.00	0.00
# feature comb.	690.90	65.59	309.16	54.98
# center numb.	8.64	1.52	21.98	4.32

C. Overall Improvements

Finally, the improvements to classification rate (or lift factor) and runtime of the new approach are outlined by means of a comparison to own, earlier approaches. Results achieved with earlier approaches for evolutionary optimization of RBF networks exist for the DM and the ID data sets [81]–[83]. A direct comparison does not make sense because previous results were obtained by another implementation running on slower computers, and—due to a high runtime—repetitions of experiments were not carried out. Therefore, we intend to imitate the old approaches with our new implementation using appropriate parameterizations which are as close as possible to the old approaches. In particular, almost all of the innovative extensions investigated above are not used and Backpropagation (DM data set) or RPROP (ID data set) are applied instead of SCG.

For the DM data set (Table XI, old approach DM-3, new approach DM-1) it can be stated that results for test data can be improved (lift factor is 22.8% higher) while the runtime is reduced significantly (by 95.3%). The network structures are smaller, and more than twice as many different solutions are investigated in less than 5% of the time needed before.

For an ID data set, the improvements are even more impressive. Table XII shows the results for the new (ID-P-1) and the old approach (ID-P-4). Runtime is reduced by about 99.1%. At the same time, the test error is decreased by about 86.0%. Again, the number of trained networks is lower but a lot more different

TABLE XII
OVERALL IMPROVEMENTS—INTRUSION DETECTION

Experiment No.	ID-P-1		ID-P-4	
Best Networks				
	μ	σ	μ	σ
$\mathcal{E}$ vali in %	0.54	0.37	3.19	3.16
$\mathcal{E}$ test in %	0.34	0.19	2.42	2.54
# centers	9.66	6.33	3.78	1.85
# features	7.22	2.98	4.36	1.56
Behavior of the EA				
	μ	σ	μ	σ
runtime in h	0.57	0.27	61.11	9.97
# trained netw.	2 734.44	152.26	6 100.00	0.00
# feature comb.	276.06	90.06	57.40	15.30
# center numb.	23.48	5.14	6.64	1.43

solutions are investigated in less than 1% of the time used previously.

VI. CONCLUSION

The main objective of our work described in this article was to make evolutionary optimization of RBF network architectures (feature and model selection) applicable to a wide range of data mining problems (in particular, classification problems). Therefore, the overall runtime of the EA had to be reduced substantially. We decided to optimize the most important architecture parameters only and to use standard techniques for representation, selection, and reproduction. Each genotype produced during evolution describes a valid solution of the optimization problem and each solution can be reached from any other point in the search space. Runtime reduction as well as improvements to classification rates or lift factors are achieved by a combination of various techniques, in particular fast fitness evaluation (hybrid training, lazy evaluation), integration of soft constraints (side conditions), and temperature-based control of the EA (to perform the trade-off between fine tuning and coarse tuning of the optimization process). With side conditions, smaller networks can be preferred which increases the interpretability of the solutions found and helps to reduce costs (computational and monetary). In practice, not all side conditions are needed for every application.

The benefits and properties of this approach were set out by means of four real-world data mining examples. It is essential to note that a large number of additional experiments were carried out. So far, problems with up to approximately 350 features, approximately 80 000 samples, and/or 19 classes were investigated. The most recent project deals with defect classification of silicon wafers. A comparison of the evolutionary approach to filter-based feature selection mechanisms and heuristic model selection techniques was beyond the scope of this article but will certainly be done in the future.

At the moment, the application of the EA requires some experience to achieve good results, i.e., the best parameter settings have to be found manually. On the other hand, the EA is quite robust against variations of parameter values. For example, we could use the same parameter settings for all intrusion and signature data sets. In the future, we will adapt parameters of the EA automatically. Additional ideas are as follows.

1) Evaluation of individuals: In order to cope with huge data sets, training, validation, and test sets will be chosen dynamically from an overall data set.
2) Soft constraints (side conditions): Other penalty models will be investigated (e.g., exponential functions) and new constraints will be introduced (e.g., constraints concerning individual costs for each feature).
3) Temperature-based control: A mechanism will be developed that detects and considers stagnation of the evolutionary optimization.

We also have to evaluate three techniques which are already implemented but not investigated here:

1) the initialization of the population by means of a complete search in a subspace of the solution space (e.g., any combination with up to a given number of input features is tested);
2) the construction of features (e.g., by means of principal component analysis);
3) the application of a Lamarckian training approach (cf. [80]), i.e., the initialization of centers and radii of descendants with values inherited from parents (if feature vector and number of centers are identical).

We also intend to investigate the automation of feature and model selection for support vector machines.

ACKNOWLEDGMENT

The authors would like to thank all of the colleagues who worked on the various application examples, in particular D. Arndt and P. Neumann (direct marketing), C. Gruber and M. Gürster (signature verification), A. Hofmann and T. Horeis (intrusion detection), and M. Bauer and R. Kern (process optimization). They would also like to thank the reviewer's for their valuable remarks, as well as R. Hoffmann for his aid.

REFERENCES

[1] S. Haykin, *Neural Networks—A Comprehensive Foundation*. New York: Macmillan, 1994.
[2] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," Artificial Intelligence Lab., Center for Biological Information Processing, Mass. Inst. Technol., Cambridge, A.I. Memo 1140, C.B.I.P. Paper 31, 1989.
[3] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, pp. 281–294, 1989.
[4] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–355, 1988.
[5] M. J. Embrechts, B. Szymanski, and K. Sternickel, "Introduction to scientific data mining: Direct kernel methods & applications," in *Computationally Intelligent Hybrid Systems: The Fusion of Soft Computing and Hard Computing*, S. J. Ovaska, Ed. New York: Wiley, 2004, ch. 10, pp. 317–362.
[6] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[7] T. Bäck, "Evolutionary algorithms in theory and practice," Ph.D. dissertation, Univ. Dortmund, Dortmund, Germany, 1994.
[8] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. New York: Springer-Verlag, 1996.
[9] H. Liu and H. Motoda, Eds., *Feature Extraction, Construction, and Selection: A Data Mining Perspective*. Boston, MA: Kluwer, 1998.
[10] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. Boston, MA: Kluwer, 1998.
[11] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 1995.
[12] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Netw.*, vol. 4, no. 5, pp. 740–747, Oct. 1993.
[13] B. Sick, "Technische Anwendungen von Soft-Computing Methoden," Univ. Passau, Passau, Germany, Lecture Notes, 2001.

[14] *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, A. A. Freitas, Ed., Springer-Verlag, New York, 2002.
[15] J. D. Schaffer, D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *Proc. Int. Workshop Combinations of Genetic Algorithms and Neural Networks (COGANN)*, Los Alamitos, NM, 1992, pp. 1–37.
[16] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep. 1999.
[17] B. Carse, A. G. Pipe, T. C. Fogarty, and T. Hill, "Evolving radial basis function neural networks using a genetic algorithm," in *Proc. IEEE Int. Conf. Evolutionary Computation*, vol. 1, Perth, Australia, 1995, pp. 300–305.
[18] G. G. Yen and H. Lu, "Hierarchical rank density genetic algorithm for radial-basis function neural network design," in *Proc. Congr. Evolutionary Computation (CEC)*, vol. 1, Honolulu, HI, 2002, pp. 25–30.
[19] Y. M. Enab, "Genetic algorithm for identifying self-generating radial basis neural networks," in *Proc. 4th Int. Conf. Artificial Neural Networks*, Cambridge, U.K., 1995, pp. 65–70.
[20] L. Guo, D.-S. Huang, and W. Zhao, "Combining genetic optimization with hybrid learning algorithm for radial basis function neural networks," *Electron. Lett.*, vol. 39, no. 22, pp. 1600–1601, 2003.
[21] M. Arakawa, H. Nakayama, Y. B. Yun, and H. Ishikawa, "Optimum design using radial basis function networks by adaptive range genetic algorithms (determination of radius in radial basis function networks)," in *Proc. Annual Conf. IEEE Industrial Electronics Soc.*, vol. 2, Nagoya, Japan, 2000, pp. 1219–1224.
[22] W. Zhao, D.-S. Huang, and L. Guo, "Optimizing radial basis probabilistic neural networks using recursive orthogonal least squares algorithms combined with micro-genetic algorithms," in *Proc. Int. Joint Conf. Neural Networks*, vol. 3, Portland, OR, 2003, pp. 2277–2282.
[23] B. A. Whitehead and T. D. Choate, "Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction," *IEEE Trans. Neural Netw.*, vol. 7, no. 4, pp. 869–880, Aug. 1996.
[24] N. Chaiyaratana and A. M. S. Zalzala, "Hybridization of neural networks and a genetic algorithm for friction compensation," in *Proc. 2000 Congr. Evolutionary Computation (CEC)*, vol. 1, La Jolla, CA, 2000, pp. 22–29.
[25] N. Chaiyaratana and A. Zalzala, "Evolving hybrid RBF-MLP networks using combined genetic/unsupervised/supervised learning," in *Proc. UKACC Int. Conf. Control*, vol. 1, Swansea, U.K., 1998, pp. 330–335.
[26] S. Chen, Y. Wu, and K. Alkadhimi, "A two-layer learning method for radial basis function networks using combined genetic and regularised OLS algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, Sheffield, U.K., 1995, pp. 245–249.
[27] S. Chen, Y. Wu, and B. L. Luk, "Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1239–1243, Oct. 1999.
[28] B. Whitehead, "Genetic evolution of radial basis function coverage using orthogonal niches," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1525–1528, Dec. 1996.
[29] W. Zhao, D.-S. Huang, and G. Yunjian, "Application of genetic algorithms to the structure optimization of radial basis probabilistic neural networks," in *Proc. 6th Int. Conf. Signal Processing*, vol. 2, Beijing, China, 2002, pp. 1243–1246.
[30] A. d. M. S. Barreto, H. J. C. Barbosa, and N. F. F. Ebecken, "Growing compact RBF networks using a genetic algorithm," in *Proc. VII Brazilian Symp. Neural Networks (SBRN)*, Recife, Brazil, 2002, pp. 61–66.
[31] W. Zhao, D.-S. Huang, and G. Yunjian, "The structure optimization of radial basis probabilistic neural networks based on genetic algorithms," in *Proc. 2002 Int. Joint Conf. Neural Networks (IJCNN)*, vol. 2, Honolulu, HI, 2002, pp. 1086–1091.
[32] S.-J. Kim, J.-S. Kim, J.-Y. Seo, H.-C. Cho, and H.-T. Jeon, "Optimal initial structure of the RBF networks using time-frequency localization and genetic algorithm," in *Proc. Congr. Evolutionary Computation (CEC)*, vol. 2, Honolulu, HI, 2002, pp. 1964–1969.
[33] O. Ciftcioglu, "GA with orthogonal transformation for RBFN configuration," in *Proc. Congr. Evolutionary Computation (CEC)*, vol. 2, Honolulu, HI, 2002, pp. 1934–1939.
[34] X. Fu and L. Wang, "Rule extraction from an RBF classifier based on class-dependent features," in *Proc. Congr. Evolutionary Computation (CEC)*, vol. 2, Honolulu, HI, 2002, pp. 1916–1921.
[35] ——, "A GA-based novel RBF classifier with class-dependent features," in *Proc. Congr. Evolutionary Computation (CEC)*, vol. 2, Honolulu, HI, 2002, pp. 1890–1894.
[36] M. W. Hwang, M. H. Kim, and J. Y. Choi, "Second-order multilayer perceptrons and its optimization with genetic algorithms," in *Proc. Congr. Evolutionary Computation (CEC)*, vol. 1, La Jolla, CA, 2000, pp. 652–658.
[37] M. W. Mak and K. W. Cho, "Genetic evolution of radial basis function centers for pattern classification," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, vol. 1, Anchorage, AK, 1998, pp. 669–673.
[38] C. Bhumireddy and C. L. P. Chen, "Genetic learning of functional link networks," in *Proc. Int. Joint Conf. Neural Networks*, vol. 1, Portland, OR, 2003, pp. 432–437.
[39] Y. Xicai, Y. Datian, and L. Ming, "Text-independent speaker identification by genetic clustering radial basis function neural network," in *Proc. 23rd Annu. Int. Conf. IEEE Engineering in Medicine and Biology Society*, vol. 2, Istanbul, Turkey, 2001, pp. 1777–1780.
[40] G. P. Liu and V. Kadirkamanathan, "Learning with multi-objective criteria," in *Proc. 4th Int. Conf. Artificial Neural Networks*, Cambridge, U.K., 1995, pp. 53–58.
[41] S. Aiguo and L. Jiren, "Evolving Gaussian RBF network for nonlinear time series modeling and prediction," *Electron. Lett.*, vol. 34, no. 12, pp. 1241–1243, 1998.
[42] Y. Bai and L. Zhang, "Genetic algorithm based self-growing training for RBF neural networks," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, vol. 1, Honolulu, HI, 2002, pp. 840–845.
[43] X. Fu and L. Wang, "Rule extraction by genetic algorithms based on a simplified RBF neural network," in *Proc. Congr. Evolutionary Computation (CEC)*, vol. 2, Seoul, Korea, 2001, pp. 753–758.
[44] N. Chaiyaratana and K. Boonlong, "Further investigations on friction compensation using a neuro-genetic based hybrid framework," in *Proc. Joint 9th IFSA World Congr. 20th NAFIPS Int. Conf.*, vol. 5, Vancouver, BC, Canada, 2001, pp. 2772–2777.
[45] E. G. M. de Lacerda, A. C. P. L. F. de Carvalho, and T. B. Ludermir, "Evolutionary optimization of RBF networks," in *Proc. 6th Brazilian Symp. Neural Networks*, Rio de Janeiro, Brazil, 2000, pp. 219–224.
[46] L. Zhang, Y. Bai, and A. Al-Amoudi, "GA-RBF neural network based maximum power point tracking for grid-connected photovoltaic systems," in *Int. Conf. Power Electronics, Machines and Drives*, Bath, U.K., 2002, pp. 18–23.
[47] J. F. D. Addison, S. Wermter, and J. MacIntyre, "Effectiveness of feature extraction in neural network architectures for novelty detection," in *Proc. 9th Int. Conf. Artificial Neural Networks (ICANN) Incorporating the IEE Conf. Artificial Neural Networks*, vol. 2, Edinburgh, U.K., 1999, pp. 976–981.
[48] P. M. Ferreira, A. E. Ruano, and C. M. Fonseca, "Genetic assisted selection of RBF model structures for greenhouse inside air temperature prediction," in *Proc. IEEE Conf. Control Applications (CCA)*, vol. 1, Istanbul, Turkey, 2003, pp. 576–581.
[49] A. M. S. Zalzala and N. Chaiyaratana, "Myoelectric signal classification using evolutionary hybrid RBF-MLP networks," in *Proc. Congr. Evolutionary Computation (CEC)*, vol. 1, La Jolla, CA, 2000, pp. 691–698.
[50] H. Leung, N. Dubash, and N. Xie, "Detection of small objects in clutter using a GA-RBF neural network," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 38, no. 1, pp. 98–118, Jan. 2002.
[51] T. L. Seng, M. Bin Khalid, and R. Yusof, "Tuning of a neuro-fuzzy controller by genetic algorithm," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 2, pp. 226–236, Apr. 1999.
[52] G. P. Liu and V. Kadirkamanathan, "Multiobjective criteria for neural network structure selection and identification of nonlinear systems using genetic algorithms," *Proc. Inst. Elect. Eng.*, vol. 146, no. 5, pp. 373–382, 1999.
[53] Q. Zhang, X. He, and J. Liu, "RBF network based on genetic algorithm optimization for nonlinear time series prediction," in *Proc. Int. Symp. Circuits and Systems (ISCAS)*, vol. 5, Bangkok, Thailand, 2003, pp. V–693–V–696.
[54] E. G. M. de Lacerda, A. C. P. L. F. de Carvalho, and T. B. Ludermir, "A study of cross-validation and bootstrap as objective functions for genetic algorithms," in *Proc. 7th Brazilian Symp. Neural Networks (SBRN)*, Recife, Brazil, 2002, pp. 118–123.
[55] H. Lin and K. Yamashita, "Hybrid simplex genetic algorithm for blind equalization using RBF networks," in *Proc. Int. Conf. Systems, Man, Cybernetics*, vol. 1, Nashville, TN, 2000, pp. 411–415.
[56] K. Najarian, G. A. Dumont, and M. S. Davies, "A learning-theory-based training algorithm for variable-structure dynamic neural modeling," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, vol. 1, Washington, DC, 1999, pp. 477–482.
[57] M. W. Hwang, J. Y. Choi, and J. Park, "Evolutionary projection neural networks," in *Proc. IEEE Int. Conf. Evolutionary Computation*, Indianapolis, IN, 1997, pp. 667–671.

[58] E. P. Maillard and D. Gueriot, "RBF neural network, basis functions and genetic algorithm," in *Proc. Int. Conf. Neural Networks*, vol. 4, Houston, TX, 1997, pp. 2187–2192.

[59] P. J. Angeline, "Evolving basis functions with dynamic receptive fields," in *Proc. IEEE Int. Conf. Systems, Man, Cybernetics Computational Cybernetics and Simulation*, vol. 5, Orlando, FL, 1997, pp. 4109–4114.

[60] L. E. Kuo and S. S. Melsheimer, "Using genetic algorithms to estimate the optimum width parameter in radial basis function networks," in *Proc. American Control Conf.*, vol. 2, Baltimore, MD, 1994, pp. 1368–1372.

[61] S. Mishra, P. K. Dash, P. K. Hota, and M. Tripathy, "Genetically optimized neuro-fuzzy IPFC for damping modal oscillations of power system," *IEEE Trans. Power Syst.*, vol. 17, no. 4, pp. 1140–1147, Nov. 2002.

[62] J. González, I. Rojas, J. Ortega, H. Pomares, F. Fernández, and A. F. Diaz, "Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1478–1495, Dec. 2003.

[63] B. A. Whitehead and T. D. Choate, "Evolving space-filling curves to distribute radial basis functions over an input space," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 15–23, Feb. 1994.

[64] V. Pariyapong and M. Parnichkun, "Ensemble structure of multiple local sensor fusion machine using evolutional pruning technique (an application to heading and rate of turn estimation)," in *Proc. IEEE Int. Conf. Industrial Technology (ICIT)*, vol. 1, Bangkok, Thailand, 2002, pp. 421–426.

[65] C. Aouiti, A. M. Alimi, and A. Maalej, "A comparative study between real and discrete genetic algorithms for the design of beta basis function neural networks," presented at the IEEE Int. Conf. Systems, Man, Cybernetics, vol. 3, Hammamet, Tunisia, 2002.

[66] S. Moalla, A. M. Alimi, and N. Derbel, "Design of beta neural systems using differential evolution," presented at the Proc. IEEE Int. Conf. Systems, Man, Cybernetics, vol. 3, Hammamet, Tunisia, 2002.

[67] Q. F. Zhao, O. Hammami, K. Kuroda, and K. Saito, "Cooperative co-evolutionary algorithm—How to evaluate a module?," in *Proc. IEEE Symp. Combinations of Evolutionary Computation and Neural Networks*, San Antonio, TX, 2000, pp. 150–157.

[68] E. Lacerda, A. de Carvalho, and T. Ludermir, *Radial Basis Function Networks 1—Recent Developments in Theory and Applications*. ser. Studies in fuzzyness and soft computing, R. J. Howlett and L. C. Jain, Eds. New York: Physica-Verlag, 2001, vol. 66, ch. 11, Evolutionary Optimization of RBF Networks, pp. 281–309.

[69] A. Topchy, O. Lebedko, V. Miagkikh, and N. Kasabov, "Adaptive training of radial basis function networks training based on cooperative evolution and evolutionary programming," in *Proc. Int. Conf. Neural Information Processing (ICONIP)*, Dunedin, New Zealand, 1997, pp. 253–258.

[70] S. A. Billings and G. L. Zheng, "Radial basis function network configuration using genetic algorithms," *Neural Netw.*, vol. 8, no. 6, pp. 877–890, 1998.

[71] L. N. de Castro and F. J. von Zuben, "Automatic determination of radial basis functions: An immunity-based approach," *Int. J. Neural Syst.*, vol. 11, no. 6, pp. 523–535, 2002.

[72] J. F. D. Addison, S. Wermter, and G. Z. Arevian, "A comparison of feature extraction and selection techniques," in *Proc. Int. Conf. Artificial Neural Networks (ICANN)*, Istanbul, Turkey, 2003, pp. 212–215.

[73] L. I. Kuncheva, "Initializing of an RBF network by a genetic algorithm," *Neurocomputing*, vol. 14, no. 3, pp. 273–288, 1997.

[74] S.-J. Kim, Y.-T. Kim, J.-Y. Ko, and H.-T. Jeon, "Optimization for the initial designed structure by localization using genetic algorithm," in *Proc. Int. Conf. Circuits/Systems, Computers and Communication (ITC-CSCC)*, Phuket, Thailand, 2002, pp. 1650–1653.

[75] V. M. Rivas, P. A. Castillo, and J. J. Merelo, "Evolving RBF neural networks," in *Proc. 6th Int. Work Conf. Artificial and Natural Neural Networks (IWANN) Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*, vol. 1, Granada, Spain, 2001, pp. 506–513.

[76] J. Gonzáles, I. Rojas, H. Pomares, and M. Salmerón, "Expert mutation operators for the evolution of radial basis function neural networks," in *Proc. 6th Int. Work Conf. Artificial and Natural Neural Networks (IWANN) Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*, vol. 1, Granada, Spain, 2001, pp. 538–545.

[77] A. J. Rivera, J. Ortega, I. Rojas, and A. Prieto, "Optimizing RBF networks with cooperative/competitive evolution of units and fuzzy rules," in *Proc. 6th Int. Work Conf. Artificial and Natural Neural Networks (IWANN) Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence*, vol. 1, Granada, Spain, 2001, pp. 570–578.

[78] B. Carse and T. C. Fogarty, "Tackling the "curse of dimensionality" of radial basis functional neural networks using a genetic algorithm," in *Proc. 4th Int. Conf. Parallel Problem Solving from Nature, Int. Conf. Evolutionary Computation*, Berlin, Germany, 1996, pp. 710–719.

[79] ——, "Fast evolutionary learning of minimal radial basis function neural networks using a genetic algorithm," in *Proc. Evolutionary Computing, AISB Workshop*, Brighton, U.K., 1996, pp. 1–22.

[80] H. Braun, *Neuronale Netze: Optimierung Durch Lernen und Evolution*. New York: Springer-Verlag, 1997.

[81] A. Hofmann and B. Sick, "Evolutionary optimization of radial basis function networks for intrusion detection," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, vol. 1, Portland, OR, 2003, pp. 415–420.

[82] P. Neumann, B. Sick, D. Arndt, and W. Gersten, "Evolutionary optimization of RBF network architectures in a direct marketing application," in *Proc. Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP)*, Istanbul, Turkey, 2003, pp. 307–314.

[83] P. Neumann, D. Arndt, and B. Sick, "An application of evolutionary and neural data mining techniques to customer relationship management," in *New Generation of Data Mining Applications*, M. M. Kantardzic and J. Zurada, Eds. Piscataway, NJ: IEEE Press, 2005, ch. 4.

[84] Y. Jin, W. von Seelen, and B. Sendhoff, "Extracting interpretable fuzzy rules from RBF neural networks," Inst. für Neuroinformatik (INF), Ruhr-Univ. Bochum, Bochum, Germany, Int. Rep. 2000–02, 2000.

[85] O. Buchtala, A. Hofmann, and B. Sick, "Fast and efficient training of RBF networks," in *Proc. Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP)*, Istanbul, Turkey, 2003, pp. 43–51.

[86] O. Buchtala, P. Neumann, and B. Sick, "A strategy for an efficient training of radial basis function networks for classification applications," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, vol. 2, Portland, OR, 2003, pp. 1025–1030.

[87] F. Schwenker, H. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Netw.*, vol. 14, no. 4–5, pp. 439–458, 2001.

[88] D. Kalyanmoy and S. Agrawal, "Understanding interactions among genetic algorithm parameters," in *Foundations of Genetic Algorithms 5*, W. Banzhaf and C. Reeves, Eds. San Mateo, CA: Morgan Kaufmann, 1999, vol. 5, pp. 265–286.

[89] V. Maniezzo, "Genetic evolution of the topology and weight distribution of neural networks," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 39–53, Feb. 1994.

[90] S. G. Roberts and M. Turega, "Evolving neural network structures: An evaluation of encoding techniques," in *Artificial Neural Nets and Genetic Algorithms*, D. W. Pearson, N. C. Steele, and R. F. Albrecht, Eds. New York: Springer-Verlag, 1995, pp. 96–99.

[91] X. Yao, "Evolutionary artificial neural networks," *Int. J. Neural Syst.*, vol. 4, no. 3, pp. 203–222, 1993.

[92] P. Köhn, "Genetic encoding strategies for neural networks," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, vol. 2, Granada, Spain, 1996, pp. 947–950.

[93] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York: Wiley, 2001.

[94] D. Arndt and W. Gersten, "Lift—A leading measure for predictive modeling," in *Proc. Int. Conf. Machine Learning and Applications (ICMLA)*, Las Vegas, NV, 2002, pp. 249–255.

[95] G. Piatetsky-Shapiro and S. Steingold, "Measuring lift quality in database marketing," *SIGKDD Explorations*, vol. 2, no. 2, pp. 76–80, 2000.

[96] H. Pohlheim, *Evolutionäre Algorithmen: Verfahren, Operatoren und Hinweise für die Praxis*. Berlin, Germany: Springer-Verlag, 2000.

[97] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Mathematical Statistics and Probability*, vol. 1, L. M. LeCam and J. Neyman, Eds., Berkeley, CA, 1967, pp. 281–297.

[98] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[99] Å. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia, PA: SIAM, 1996.

[100] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore, MA: Johns Hopkins Univ. Press, 1996.

[101] M. F. Møller, "A scaled conjugate algorithm for fast supervised learning," *Neural Netw.*, vol. 6, no. 1, pp. 525–533, 1993.

[102] A. J. Shepherd, *Second-Order Methods for Neural Networks*. London, U.K.: Springer-Verlag, 1997.

[103] M. Klimek, "Evolutionäre Architekturoptimierung von RBF-Netzen," M.S. thesis, Univ. Passau, Passau, Germany, 2003.

[104] L. Prechelt, "PROBEN 1—A set of neural network benchmark problems and benchmarking rules," Univ. Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21/94, 1994.

[105] C. L. Blake and C. J. Merz. (1998) UCI repository of machine learning databases. [Online]. Available: http://www.ics.uci.edu/~mlearn/ML-Repository.html

[106] A. Wespi, G. Vigna, and L. Deri, Eds., *Recent Advances in Intrusion Detection*. New York: Springer, 2002.
[107] S. Northcutt and J. Novak, *Network Intrusion Detection*, 3rd ed. Indianapolis, IN: New Riders, 2002.
[108] R. Durst, T. Champion, B. Witten, E. Miller, and L. Spagnuolo, "Testing and evaluating computer intrusion detection systems," *Commun. ACM*, vol. 42, pp. 53–61, 1999.
[109] R. P. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation," *Comput. Netw.*, vol. 34, no. 4, pp. 579–595, 2000.
[110] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in *Proc. DARPA Information Survivability Conf. and Exposition (DISCEX)*, vol. 2, Hilton Head, SC, 2000, pp. 12–26.
[111] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, vol. 2, Honolulu, HI, 2002, pp. 1702–1707.
[112] W. Lee, S. J. Stolfo, and K. W. Mok, "A data mining framework for building intrusion detection models," in *Proc. IEEE Symp. Security and Privacy*, Oakland, CA, 1999, pp. 120–132.
[113] B. Pfahringer, "Winning the KDD99 classification cup: Bagged boosting," *SIGKDD Explorations*, vol. 1, no. 2, pp. 65–66, 2000.
[114] A. Hofmann, C. Schmitz, and B. Sick, "Intrusion detection in computer networks with neural and fuzzy classifiers," in *Proc. Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP)*, Istanbul, Turkey, 2003, pp. 316–324.
[115] ——, "Rule extraction from neural networks for intrusion detection in computer networks," in *Proc. IEEE Int. Conf. Systems, Man, Cybernetics*, vol. 2, Washington, DC, 2003, pp. 1259–1265.
[116] V. S. Nalwa, "Automatic on-line signature verification," in *Biometrics: Personal Identification in Networked Society*, A. K. Jain, R. Bolle, and S. Panhanti, Eds. Boston, MA: Kluwer, 1999.
[117] J. Ashbourn, *Biometrics: Advanced Identity Verification: The Complete Guide*. London, U.K.: Springer-Verlag, 2000.
[118] C. Hook, J. Kempf, and G. Scharfenberg, "New pen device for biometrical 3d pressure analysis of handwritten charakters, words and signatures," in *Proc. ACM Workshop Biometrics: Methods and Applications (WBMA)*, Berkeley, CA, 2003, pp. 38–44.
[119] O. Rohlík, P. Mautner, V. Matousek, and J. Kempf, "HMM based handwritten text recognition using biometrical data acquisition pen," in *Proc. IEEE Int. Symp. Computational Intelligence in Robotics and Automation*, vol. 2, Kobe, Japan, 2003, pp. 950–953.
[120] P. Mautner, O. Rohlík, V. Matousek, and J. Kempf, "Signature verification using ART-2 neural network," in *Proc. 9th Int. Conf. Neural Information Processing (ICONIP)*, vol. 2, Singapore, 2002, pp. 636–639.
[121] ——, "Fast signature verification without a special tablet," presented at the 9th Int. Workshop Systems, Signals and Image Processing (IWSSIP), Manchester, U.K..
[122] P. Mautner, V. Matousek, O. Rohlík, and J. Kempf, "Signature verification using unsupervised learned neural networks," in *Proc. Int. Workshop Artificial Neural Networks in Pattern Recognition (ANNPR)*, Florence, Italy, 2003, pp. 879–885.
[123] C. Gruber, J. Kempf, G. Scharfenberg, and B. Sick, "On-Line Signature Verification with a Biometric Pen using a Combination of Static and Dynamic Classifiers," 2005, to be published.
[124] D. Arndt and W. Gersten, "External data selection for data mining in direct marketing," in *Proc. Int. Conf. Information Quality*, Boston, MA, 2001, pp. 44–61.
[125] W. Gersten, R. Wirth, and D. Arndt, "Predictive modeling in automotive direct marketing: Tools, experiences and open issues," in *Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Boston, MA, 2000, pp. 398–406.
[126] P. Neumann, "Soft-computing methods for the identification of relevant features and the prediction of customer behavior in the automotive industry," M.S. thesis, Univ. Passau, Passau, Germany, 2002.
[127] *Wacker Siltronic—Wafers for the World of Microchips*. Burghausen, Germany: Wacker Siltronic AG, 2000.
[128] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE Int. Conf. Neural Networks*, San Francisco, CA, 1993, pp. 586–591.

Oliver Buchtala is currently studying computer science at the University of Passau, Passau, Germany, where he is also working as a Student Assistant at the Institute for Computer Architectures.

His interests include soft computing and image processing.

Mr. Buchtala was awarded a travel grant from the IEEE Neural Network Society for a paper at the IJCNN 2003 in Portland, OR.

Manuel Klimek received the diploma in computer science from the University of Passau, Passau, Germany, in 2003.

Currently, he is a Software Developer for payment devices at the EL-ME AG, Germany. Apart from soft computing, his interests include image processing, software engineering, and compilers. He actively participates in the open source community.

Mr. Klimek was awarded the Prize of the Chamber of Commerce and Industry Passau, Germany, in 2004.

Bernhard Sick (M'02) received the diploma, the Ph.D. degree, and the postdoctoral lecture qualification (Habilitation) in computer science from the University of Passau, Passau, Germany, in 1992, 1999, and 2004, respectively.

Currently, he is an Assistant Professor at the University of Passau. He is conducting research and development in the areas of theory and application of soft computing techniques, fusion of soft computing and hard computing methods, specification and development of real-time systems, and the development of frameworks for component-based, visual software programming for embedded systems. He has authored more than 50 peer-reviewed publications in these areas. He is a member of Gesellschaft fuer Informatik (GI).