Multiobjective Evolutionary Optimization of the Size, Shape, and Position Parameters of Radial Basis Function Networks for Function Approximation

Jesús González, Ignacio Rojas, Julio Ortega, *Member, IEEE*, Héctor Pomares, Fco. Javier Fernández, and Antonio Fco. Díaz

***Abstract*—This paper presents a multiobjective evolutionary algorithm to optimize radial basis function neural networks (RBFNNs) in order to approach target functions from a set of input-output pairs. The procedure allows the application of heuristics to improve the solution of the problem at hand by including some new genetic operators in the evolutionary process. These new operators are based on two well-known matrix transformations: singular value decomposition (*SVD*) and orthogonal least squares (*OLS*), which have been used to define new mutation operators that produce local or global modifications in the radial basis functions (RBFs) of the networks (the individuals in the population in the evolutionary procedure). After analyzing the efficiency of the different operators, we have shown that the global mutation operators yield an improved procedure to adjust the parameters of the RBFNNs.**

***Index Terms*—Evolutionary computation, neural networks, radial basis functions (RBFs), orthogonal transformations, heuristics.**

I. Introduction

RADIAL BASIS function neural networks (RBFNNs) [8] consist of neurons which are locally tuned. An RBFNN can be regarded as a feedforward artificial neural network (ANN) [37] with a single layer of hidden units, whose responses are the output of radial basis functions (RBFs), as shown in Fig. 1. Formally, an RBFNN can be described by the following equation

$$\mathcal{F}(\boldsymbol{x};\Phi,\Omega)=\sum_{j=1}^{m}\omega_j\cdot\phi_j(\boldsymbol{x};\boldsymbol{c}^j,r^j) \tag{1}$$

where d is the input space dimensionality and m is the number of hidden units. The output of the net $\mathcal{F}$ depends on the input vector $\boldsymbol{x}=(x_1,\ldots,x_d)^T$, and on the sets of RBFs $\Phi=\{\varphi_1,\ldots,\varphi_m\}$ and weights $\Omega=\{w_1,\ldots,w_m\}$, as explicitly indicated in (1). Each RBF locally contributes to the net output. This contribution will depend on the distance from the input vector $\boldsymbol{x}$ to the center $\boldsymbol{c}^j$ of each RBF ϕ_j, and is 1 if $\|\boldsymbol{x}-\boldsymbol{c}^j\|=0$ and tends to 0 as $\|\boldsymbol{x}-\boldsymbol{c}^j\|$ increases. There are many possibilities to choose

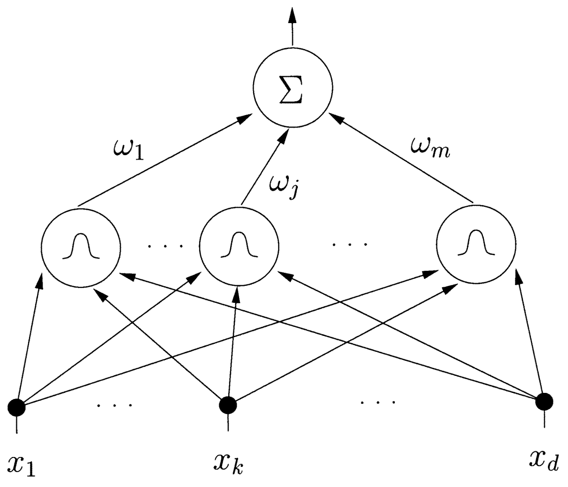

Fig. 1. Structure of a RBFNN.

RBFs that satisfy this condition [47], [69], [70], [81], although the most commonly used RBF is the Gaussian function

$$\phi_j(\boldsymbol{x};\boldsymbol{c}^j,r^j)=\exp\left(\frac{\|\boldsymbol{x}-\boldsymbol{c}^j\|^2}{(r^j)^2}\right) \tag{2}$$

where $\boldsymbol{c}^j$ is the RBF center and determines where it is allocated in the input space, and r^j is its radius, which indicates how the basis function response decreases as the input vector $\boldsymbol{x}$ recedes from $\boldsymbol{c}^j$.

RBFNNs as presented earlier have been shown to be universal approximators [61], [62] even when using the same radius for all the RBFs in the net, although if a different radius r^j is used for each RBF ϕ_j, it is possible to obtain a better fitted model using fewer hidden units [59].

Due to their simple structure, compared with other ANNs such as multilayer perceptrons (MLPs) [38], there has been increasing research interest in RBFNNs and their applicability as function approximators, plant controllers, and classifiers in recent years [44], [47], [51], [71], [69], [70], [68]. The optimum values for the weights in Ω can be obtained by solving an overdetermined linear system of equations once the RBFs in Φ have been fixed [5], [57]. There are several well known methods to perform this task, such as Cholesky decomposition [67], [66], the orthogonal least squares (OLS) method [15], or singular value decomposition (SVD) [45]. Thus, the problem can be reformulated as one of how to set

Manuscript received February 26, 2002; revised August 22, 2002 and March 3, 2003. This work was supported in part by the Spanish Ministerio de Ciencia y Tecnología under Project DPI2001-3219.

The authors are with the Department of Computer Architecture and Computer Technology, E.T.S. Ingeniería Informáticá, University of Granada, E. 18071 Granada, Spain.

Digital Object Identifier 10.1109/TNN.2003.820657

the parameters concerning each RBF (its center and radius) in order to minimize an error criterion. In the literature, several algorithms to identify these parameters have been published, especially because one-stage gradient-descent algorithms have stability problems when dealing with spread parameters of RBFs [5], [47], [46].

One of the first, proposed in [57], consists in allocating the centers of the RBFs using a clustering technique such as the C-means algorithm [24] or the fuzzy C-means algorithm [4], estimating the radii by using heuristics like the k-nearest neighbor (KNN), the closest input vector (CIV) [47], or other heuristic criteria for two-stage RBF learning, e.g., those proposed by Bruzzone *et al.* in classification tasks [9]. These approaches are useful for classification problems, where clustering algorithms perform well. Nevertheless, the use of clustering algorithms may not produce a good initialization in the set of RBFs for a function approximation problem. In [34], based on the enhanced LEG algorithm [63], the clustering for function approximation (CFA) algorithm is proposed to obtain the initial position of the RBFs taking into account the target function output variability. Other recent works have proposed different sophisticated algorithms such as the resource allocation network (RAN) [64], or the growing radial basis function [47], which add neurons to the net until a stopping criterion is met. An enhanced version of RAN that performs node pruning based on SVD and QRcp (QR[1]. with column pivoting) decompositions has been proposed recently [75], [76]. This new algorithm allows the determination of the optimal number of RBFs in the network as well as the ideal input space configuration for time series prediction applications.

Evolutionary Algorithms (EAs) [56] have also been applied to optimize the parameters defining RBFNNs [50], [54], [83], [84]. However, EAs are a generic optimization technique whose results can be improved if some expert knowledge about the problem to be solved is incorporated [2]. This hybridization of the robustness and strength of EAs and the expertness of the heuristics improves the optimization procedure. Thus, adapted EAs obtain better results than generic ones because they can guide the search toward solutions that an expert knowledge of the problem expects to be superior. The incorporation of expert knowledge has been carried out in this paper by constructing some problem-specific mutation operators. These operators differ from the original ones in that they do not produce blind changes to the individuals they affect, but attempt to improve the mutated individual by analyzing and altering it according to some heuristics [33]. Specifically, in the problem of optimizing the parameters that define an RBFNN from training samples, there exist two good heuristics to determine the relevance of a basis function in the net output: OLS and SVD [86].

This paper presents a brief review of the orthogonal transformations OLS and SVD in Section II. Section III summarizes the concept of MultiObjective Evolutionary Algorithm (MOEA) and justifies its use. Section IV describes some expert mutation operators based on the OLS and SVD transformations specifically designed for the optimization of RBFNNs. These operators are compared with a random (blind) type mutation in Section V, and some experimental results are presented in Section VI, where the MOEA is applied to several benchmark problems and the results are compared with those obtained by other approaches used to solve function approximation problems. Finally, some interesting conclusions are presented in Section VII.

[1]QR is a widely used orthogonal matrix decomposition with Q orthogonal and R upper triangular, the reader is referred [32] for further reading.

II. Orthogonal Transformations

Once the parameters concerning the RBFs have been fixed, their associated weights can be optimally calculated using a linear systems solver. Equation (1) can be seen as a special case of linear regression

$$y^k = \sum_{j=1}^{m} p_{kj} \cdot \omega_j + e^k \quad k+1, \ldots, n \tag{3}$$

where p_{kj} is given by

$$p_{kj} = \phi(\boldsymbol{x}^k; \boldsymbol{c}^j, r^j) \tag{4}$$

In this regression model, p_{kj} are the regressors, ω_j the parameters, and e^k are the residual errors for each input vector $\boldsymbol{x}^k$. Given n input-output samples, (3) can be expressed as

$$\boldsymbol{y} = P\boldsymbol{\omega} + \boldsymbol{e} \tag{5}$$

where $\boldsymbol{y} = [y^1, \ldots, y^n]^T \in \mathbb{R}^n$ is the column vector containing all the expected outputs, $P = [\boldsymbol{p_1}, \ldots, \boldsymbol{p_m}] \in \mathbb{R}^{n\times m}$ is a matrix whose columns $\boldsymbol{p_j} = [p_{1j}, \ldots, p_{nj}]^T \in \mathbb{R}^n$ represent the output of the jth basis function for all the input vectors. The vector $\boldsymbol{e} = [e^1, \ldots, e^n]^T \in \mathbb{R}^n$ contains all the errors committed by the model (assumed to be uncorrelated), and $\boldsymbol{\omega}$ is a vector containing the net weights. Henceforth in this paper P will be the activation matrix of the RBFNN and $P\boldsymbol{\omega}$ will be the predictor of the model. The number of training samples n is usually greater than the number of RBFs m, and so we only have to solve the following overdetermined linear system

$$\boldsymbol{y} = P\boldsymbol{\omega} \tag{6}$$

to minimize the approximation error of the net and find the optimal weights (in the least squares sense). There are several ways to solve overdetermined linear systems. The following sections present two of the most commonly used methods: OLS and SVD.

A. OLS

This method was originally employed in [15] to calculate the optimum weights of an RBFNN. It also estimates the relevance of each RBF ϕ_j in the output of the net by assigning it an error reduction ratio $[\mathrm{err}]_j$. OLS transforms the columns of the activation matrix P into a set of orthogonal vectors $\boldsymbol{u_j}$. This transformation is performed by applying the Gram-Schmidt orthogonalization method [32] and produces

$$P = UR. \tag{7}$$

Note that (7) gives the same information as the "standard" QR decomposition of $P(P = QR)$ [32]. Substituting (7) in (5) we obtain

$$\boldsymbol{y} = UR\boldsymbol{\omega} + \boldsymbol{e} = U_{\boldsymbol{g}} + \boldsymbol{e} \tag{8}$$

where $\boldsymbol{g} = R\boldsymbol{\omega}$. As $\boldsymbol{u_j}$ and $\boldsymbol{u_l}$ are orthogonal $\forall_j \neq l$, the sum of squares of $y^k(\boldsymbol{y}^T\boldsymbol{y})$ can be written as

$$\boldsymbol{y}^T\boldsymbol{y} = \sum_{j=1}^{m} g_j^2 \boldsymbol{u_j}^T \boldsymbol{u_j} + \boldsymbol{e}^T\boldsymbol{e} \tag{9}$$

Dividing both sides of (9) by n, it can be seen how the model variance $(\boldsymbol{y}^T\boldsymbol{y}/n)$ is decomposed into explained and residual variances

$$\frac{\boldsymbol{y}^T\boldsymbol{y}}{n} = \frac{\sum_{j=1}^{m} g_j^2 \boldsymbol{u_j}^T \boldsymbol{u_j}}{n} + \frac{\boldsymbol{e}^T\boldsymbol{e}}{n} \tag{10}$$

So, as $g_j^2 \boldsymbol{u_j}^T \boldsymbol{u_j}/n$ is the contribution of $\boldsymbol{u_j}$ to the total output variance, we can define the error reduction ratio of $\boldsymbol{u_j}$ as [15]

$$[\mathrm{err}]_j = \frac{g_j^2 \boldsymbol{u_j}^T \boldsymbol{u_j}}{\boldsymbol{y}^T\boldsymbol{y}} \quad \forall_j = 1, \ldots, m \tag{11}$$

This ratio can be used to rank the RBFs according to their contribution to the reduction of the approximation error. If we want to keep the r most relevant RBFs $(r < m)$, we will select the r basis functions that have the highest error reduction ratios. This method can be used to prune the least relevant RBFs in the net, thus obtaining a simpler model whose approximation error is as close as possible to the error of the original net. A more detailed discussion of the OLS transformation can be found in [14]–[16], [86].

B. SVD

The SVD of the activation matrix P produces

$$P = U\Sigma V^T \tag{12}$$

where $U \in \mathbb{R}^{n\times m}$ is a matrix with orthogonal columns, $\Sigma \in \mathbb{R}^{m\times m}$ is a diagonal matrix whose elements σ_j are positive or zero and are called singular values of P, and $V \in \mathbb{R}^{m\times m}$ is an orthogonal matrix.

Each of the columns $\boldsymbol{u_j}$ in U is related to an RBF of the net, and has an associated singular value σ_j that estimates its degree of linear independence in the system. The higher σ_j is, the more linearly independent is $\boldsymbol{u_j}$ in U. So, if we only want to select the r most relevant basis functions for network pruning $(r < m)$, we only have to maintain the r RBFs with the highest associated singular values [32]

$$P_r = U_r \Sigma_r V_r^T \tag{13}$$

where Σ_r is a diagonal matrix containing the highest r singular values, and U_r and V_r keep the columns of U and V associated with the singular values in Σ_r. This is an easy method to discard the $m - r$ least relevant RBFs in the net [45], [58], [58], [85], [86].

SVD decomposition is a complicated orthogonal transformation and its implementation is beyond the scope of this paper. Here, we will only mention that there are two possible implementations, faster of which is SVD-QRcp, described in [32], [73], [74]. This implementation uses Householder transformations for faster obtention of SVD, but obtains the columns of U with an altered order, so it is necessary to apply the QR algorithm with Column Pivoting (QRcp) to get an estimation of their true order. The other method is the Kogbetliantz algorithm [48], [49], which maintains the original order of the columns, but is much slower, although easy to parallelize [7], [6], [13].

In the following section we present a multiobjective evolutionary algorithm that allows the determination of the optimal number, position and shape of RBFs for function approximation problems. In this algorithm we also introduce new genetic operators that are based on the orthogonal transformations presented in this section.

III. Multiobjective Evolutionary Algorithms

The automatic optimization of a RBFNN from training data [68], [69] is a problem in which two clearly competing objectives must be satisfied. The model's prediction error must be minimized in order to achieve a well fitted model, while the number of RBFs should be as low as possible to obtain a reliable interpolator. The problem here is how to satisfy both objectives simultaneously. Improving one of them will probably worsen the other. This kind of problem is known as a Multi-Objective Problem (MOP) [60], [40], [12], [36], and their solutions are usually sub-optimal for each objective in particular, but "acceptable" taking all the objectives into account, where "acceptable" is totally subjective and problem-dependent.

The algorithms proposed in the literature to construct RBFNNs from examples usually try to find a unique model with a compromise between its complexity and its prediction error. This is not an adequate approach. In MOPs there is usually more than one alternative optimal solution (each making different compromises between multiple objectives) that can be considered equivalent. Thus, it is very difficult to adapt conventional optimization techniques to solve MOPs because they were not designed to deal with more than one solution simultaneously [25]. Nevertheless, EAs maintain a population of potential solutions for the problem, thus making it easier to adapt them to solve MOPs [25]. In particular, the fitness of the individuals must be adapted to comprise all the objectives to be satisfied and new mutation operators must be designed to alter the structure of RBFNNs.

The great difference between single objective and multiple objective problems is that the set of solutions is not completely ordered for MOPs. Generally, in the case of a problem with n_{obj} competing objectives, each one of them measured by the objective function $f_i(i = 1, \ldots, n_{\mathrm{obj}})$, we can define a global objective function f that meets the following relations for two potential solutions for the problem ι_1 and ι_2

$$\begin{aligned} f(\iota_1) = f(\iota_2) &\Longleftrightarrow f_i(\iota_1) = f_i(\iota_2) \quad \forall_i \in 1, 2, \ldots, n_{\mathrm{obj}} \\ f(\iota_1) \leq f(\iota_2) &\Longleftrightarrow f_i(\iota_1) \leq f_i(\iota_2) \quad \forall_i \in 1, 2, \ldots, n_{\mathrm{obj}} \\ f(\iota_1) < f(\iota_2) &\Longleftrightarrow f_i(\iota_1) \leq f_i(\iota_2) \wedge (f(\iota_1) \neq f(\iota_2)). \end{aligned} \tag{14}$$

Taking into account the above relations, the Pareto-dominance criterion can be defined as

$$\begin{aligned} &\iota_1 \prec \iota_2 \ (\iota_1 \text{ dominates } \iota_2) \\ &\qquad \Longleftrightarrow f(\iota_1) < f(\iota_2) \\ &\iota_1 \preceq \iota_2 \ (\iota_1 \text{ weakly dominates } \iota_2) \\ &\qquad \Longleftrightarrow f(\iota_1) \leq f(\iota_2) \\ &\iota_1 \sim \iota_2 \ (\iota_1 \text{ is indifferent to } \iota_2) \\ &\qquad \Longleftrightarrow f(\iota_1) \not\leq f(\iota_2) \wedge f(\iota_2) \not\leq f(\iota_1) \end{aligned} \tag{15}$$

where $\iota_1 \preceq \iota_2$ means that ι_1 is a better global solution than $\iota_2, \iota_1 \preceq \iota_2$ means that ι_1 is better or equal than ι_2, and $\iota_1 \sim \iota_2$ means that ι_1 and ι_2 are not comparable solutions.

A Pareto-optimum solution [60] is defined as an individual that cannot be dominated by any one in the solution set

$$\nexists \iota_i \in D : \iota_i \prec \iota_j. \tag{16}$$

A good multiobjective algorithm should find as many Pareto-optimum solutions as possible, to provide the final user with the possibility of choosing the right solution following his own criteria.

Multi-Objective Evolutionary Algorithms (MOEAs) constitute a robust optimization technique that has been successfully applied to several optimization problems [19], [26], [27], [42], [79]. Its strength is based on its simplicity and easy implementation. In this case, the problem is to construct an RBFNN for function approximation from examples, and the two competing objectives are the net complexity (number of RBFs) and its approximation error.

We have also incorporated a fine tuning step into the algorithm to improve the precision of the solutions in the population. This step applies, in each generation, a few iterations of the Levenberg-Marquardt minimization algorithm [22], [55] to the nondominated individuals in the population. As these individuals are expected to have a higher number of copies in the next generation, the local adjustment introduced by the minimization algorithm is propagated to the whole population in a few generations [33].

After the MOEA finishes and returns a set of solutions with different compromises between their structural complexity and their approximation error, the minimization algorithm is applied to each one of them to reach the closest local optimum in the search space.

As stated earlier, the expert knowledge has been incorporated into the algorithm by constructing evolutionary operators that apply heuristic-based changes to the individuals in the population. These mutation operators are based on the SVD and OLS transformations presented in Section II.

A. Ranking the Solutions

As stated above, the set of solutions is not completely ordered in a MOP. The Pareto-dominance criterion allows comparing two different solutions, but it can not measure the difference between them. There have been proposed several approaches in the literature to overcome this problem, such as the MOGA presented in [25] or the NSGA described in [77]. In this paper we use the approach proposed in [25], which defines the concept of *rank* of a solution as

$$\operatorname{rank}\left(\iota_j^t\right) = 1 + \operatorname{dom}_j^t \tag{17}$$

where dom_j^t represents the number of individuals dominating ι_j^t in the current population. Note that rank improves when becomes smaller, that is, as the rank of ι_j^t gets a lower value, ι_j^t represents a better solution for the MOP, thus, all the Pareto-optimum solutions will be assigned a rank value of one. Fig. 2 shows an example of rank assignment.

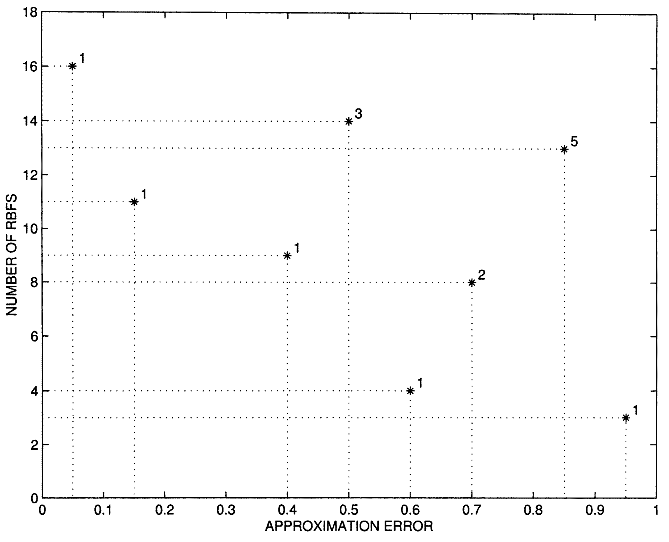

Fig. 2. Ranking of the solutions.

Once each individual has been assigned a rank, a scalar dummy fitness is obtained for all the individuals following the algorithm.

1) Rank the RBFNNs in the population according to their rank value.
2) Assign an initial dummy scalar fitness to each RBFNN by a linear interpolation between the lower and higher rank values in the population.
3) Obtain the final scalar dummy fitness of each RBFNN as the mean of the initial dummy fitness scalar values of all the RBFNNs having the same rank value.

Table I shows the execution of this algorithm for the population ranked in Fig. 2. This simple modification in the fitness evaluation of the RBFs allows a generic EA to solve an MOP transparently, that is, without changing any other of its components. As an example, the selection scheme can remain unaltered and select the solutions according to their scalar dummy fitness. To avoid a premature convergence of the algorithm a niching strategy has also been incorporated. The niching scheme used in this paper is fully discussed in [31], [21], [77].

IV. Proposed Evolutionary Operators

This section presents some new evolutionary operators specifically designed for the problem of optimizing the parameters of an RBFNN. These new operators apply random changes to the individuals they affect to maintain the diversity in the population and to provide mechanisms to escape from local minima [30], [39], but they apply different criteria.

We will present a crossover operator, together with some mutation operators that can be organized into two groups: operators to change the structure of the net and operators to adjust the parameters of the RBFNNs.

The former group contains SVD-based Pruning (SVDP), OLS based-Pruning (OLSP), and the Splitting of RBFs (SPLIT).

The latter group is composed of the Locally Random Mutation (LRM), a blind operator, and heuristically guided

TABLE I
DUMMY FITNESS ASSIGNMENT FOR THE POPULATION DEPICTED IN FIG. 2

Step	Rank and Dummy Fitness								Description
1	1	1	1	1	1	2	3	5	Sorting by Rank
2	1	1.57	2.14	2.71	3.29	3.86	4.43	5	Initial Dummy Fitness
3	2.14	2.14	2.14	2.14	2.14	3.86	4.43	5	Final Dummy Fitness

operators such as the Local OLS-based Mutation (LOLSM), the Local SVD-based Mutation (LSVDM), the Global OLS-based Mutation (GOLSM), and the Global SVD-based Mutation (GSVDM). One important characteristic, analyzed in Section V, is the scope of the changes they perform, which can be local to an RBF (LRM, LOLSM, and LSVDM), or global within the input space (GOLSM and GSVDM).

A. RBFNN Crossover

This operator takes two RBFNNs and returns two offspring, combining the genetic information from their ancestors. The offspring are generated by interchanging several RBFs in the original nets. Some RBFs are selected randomly in one of the ancestors and are replaced by the closest RBFs in the input space belonging to the second progenitor, in order to avoid the generation of two new RBFNNs that could leave input regions uncovered. After the exchange of information, the optimum weights are obtained for the descendants.

B. SVDP: SVD-Based Pruning

One of the orthogonal transformations described in Section II to obtain a measure of the relevance of each RBF is to perform the SVD of the net's activation matrix. This decomposition provides a set of singular values, one per RBF. High singular values identify important RBFs, while low singular values detect less relevant basis functions. This heuristic is well known and has been applied as a pruning criterion by some algorithms in the literature [45], [58], [85], [86].

Taking this information into account, this mutation operator assigns a pruning probability p_j to each RBF ϕ_j that is inversely proportional to its associated singular value

$$p_j = 1 - \frac{\sigma_j}{\sum_{k=1}^{m} \sigma_k}. \tag{18}$$

Less-important RBFs are assigned a greater pruning probability than more relevant ones. Once these pruning probabilities have been calculated, an RBF is randomly selected and deleted, and the optimum weights for the remaining RBFs are obtained.

C. OLSP: OLS-Based Pruning

The other orthogonal transformation described in Section II is OLS. This method also assigns a relevance value to each RBF, but with an important difference: OLS takes into account the expected output for each input vector in the training set. Thus, the relevance of each RBF is closely related to its contribution to the reduction of the training error. The RBFs making a bigger contribution to the training error reduction will be more sensitive to the pruning than those making a smaller contribution.

OLS has been widely applied as a pruning heuristic for RBFNNs and fuzzy systems [14]–[16], [86]. In this case, this orthogonal transformation has been used to guide a mutation operator to select the right RBF to be deleted. In order to minimize the effect of the mutation on the net approximation error, OLS is employed to assign a deletion probability to each RBF according to its relevance in the output. An RBF ϕ_j is assigned a pruning probability p_j that is inversely proportional to its error reduction ratio $[\mathrm{err}]_j$:

$$p_j = 1 - \frac{[\mathrm{err}]_j}{\sum_{k=1}^{m} [\mathrm{err}]_k} \tag{19}$$

After the pruning probability for each RBF is obtained, one of them is randomly deleted and the optimum weights of the new net are calculated.

D. SPLIT: Splitting of RBFs

The objective of this mutation operator is to detect the input areas that are worse modeled by the RBFNN, i.e., those with a higher approximation error, and to increase the number of RBFs in these areas in order to increase the variance of the data explained by the net. To carry out this task, the operator estimates the contribution of each RBF to the whole approximation error using the following expression

$$e^j = \sum_{k=1}^{n} \frac{\phi_j(\boldsymbol{x}^k)}{\sum_{i=1}^{m} \phi_i(\boldsymbol{x}^k)} |\mathcal{F}(\boldsymbol{x}^k) - y^k|, \quad j = 1, \ldots, m. \tag{20}$$

A high value of e^j means that the RBF ϕ_j is not able to model correctly the training data that most activate it, so, it would be desirable to increment the number of RBFs in this input zone, in order to minimize the approximation error caused by the training examples that activate ϕ_j. Thus, the mutation operator assigns a splitting probability to each RBF ϕ_j proportional to its contribution to the approximation error e^j. This means that those RBFs with a higher contribution to the approximation error will have more probability of being split. Once that all the RBFs have been assigned a splitting probability, one basis function is randomly selected according to these probabilities distribution.

After the RBF ϕ_j has been selected, the 2-means algorithm is run with the input examples that are closer to the center of ϕ_j than to any other RBF center, obtaining two new positions for two new RBFs, ϕ_{j1} and ϕ_{j2}, which will substitute ϕ_j in the affected net. The radii for ϕ_{j1} and ϕ_{j2} are calculated using the k-Nearest Neighbor (KNN) heuristic [57], [47], and the optimum weights for all the weights of the new net are obtained using the Cholesky method.

A similar splitting criterion was used in [47], where the RBFs with higher classification error were split in a similar way. The difference between the proposed criterion and the one presented in [47] is the way of measuring the error committed by each

RBF. In [47] it was an easy issue because the RBFNNs were used to solve classification problems, thus, the error committed by each RBF was the number of misclassified patterns. In a function approximation problem, this task becomes quite more difficult because all the RBFs contribute in the final output, so the approximation error has to be shared in any way between the RBFs. In this case, it has been used the sum-of-absolute-values metric, although the euclidean norm was another possibility [82]. Nevertheless, both metrics are able to assign a higher splitting probability to those RBFs that make a major contribution to the approximation error.

E. LRM: Locally Random Mutation

This operator is a direct adaptation of the classical blind genetic mutation operator [39]. All the parameters concerning the RBFs of the net have the same probability of being altered, and once a parameter is selected, it undergoes a locally random change. Due to the large number of parameters defining an RBFNN, there are several possible implementations for this operator. The chosen implementation will alter only the centers and radii of the net, because the weights can be optimally calculated once these parameters have been fixed, as described in Section II. Basically, this operator performs the following steps.

- Randomly select an RBF to be altered. All the RBFs have the same probability of being chosen.
- Decide whether to alter its radius or its center. This decision is made randomly, using a probability of 0.5 for each of the possibilities.
- Perform a local alteration of the center or the radius following the steps detailed in Sections IV-E–1 and IV-E–2, [35].
- Obtain the optimum weights for the new net.

1) Locally Random Change of the Center of an RBF: The center of an RBF is modified by applying an offset vector. The norm of this offset is smaller than the RBF radius, to preserve the relative positions of all the RBFs in the input space

$$(\boldsymbol{c}^j)' = \boldsymbol{c}^j + \boldsymbol{off}^j, \quad (|\boldsymbol{off}^j| \le r^j), j = 1, \ldots, m. \tag{21}$$

Each one of the components in the offset vector $\boldsymbol{off}$ is obtained using the following expression

$$off_i^j = \mathrm{sign}() \cdot \mathrm{u}(0, r^j), \quad i = 1, \ldots, d \tag{22}$$

where d is the input space dimensionality and $\mathrm{sign}()$ is a function defined as

$$\mathrm{sign}() = \begin{cases} 1 & \text{if } \mathrm{u}(0,1) < 0.5 \\ -1 & \text{otherwise} \end{cases} \tag{23}$$

and the function $\mathrm{u}(a, b)$ randomly returns a real number chosen uniformly in the interval $[a, b)$.

Once the offset is applied, there is a restriction that has to be checked to validate the mutation: the new center must be sufficiently close to the set of training data, so that the following expressions are satisfied

$$c_i^{j'} \ge m_i - r^j, \tag{24}$$
$$c_i^{j'} \le M_i + r^j \tag{25}$$

where m_i and M_i are defined as

$$m_i = \min_l \{x_i^l\}, \quad l = 1, \ldots, n \tag{26}$$
$$M_i = \max_l \{x_i^l\}, \quad l = 1, \ldots, n. \tag{27}$$

These constraints ensure that the new position of the RBF will allow it to be activated by some training samples. Without them, a mutation could place an RBF far from the training data, and thus this RBF would not contribute to the net output.

2) Locally Random Change of the Radius of an RBF: This alteration also applies a random offset, in this case to the radius of the RBF

$$(r^j)' = r^j + \mathrm{sign}() \cdot \mathrm{u}(0, r^j). \tag{28}$$

The expression allows the radius to change according to its norm. Once the alteration has been performed, some constraints have to be met. The first one is that the new radius must be a sufficiently large positive real number, so that the RBF can be activated

$$\textbf{IF} \quad (r^j)' < U, \quad \textbf{THEN} \quad (r^j)' = U \tag{29}$$

where U is a lower threshold for the RBF radii to avoid division by zero in (2). An RBF with such a small radius will probably not be activated, but this is not really a problem. Mutation operators have to add diversity to the population, and if they produce a bad solution, it will probably not survive in the next generation.

Another two constraints to be satisfied are

$$c_i^j + (r^j)' \ge m_i, \tag{30}$$
$$c_i^j - (r^j)' \le M_i \tag{31}$$

These restrictions force all the RBFs whose centers are far from the training data to have a sufficiently large radius to be activated by some input vectors.

F. LSVDM: Local SVD-Based Mutation

The set of singular values obtained by applying SVD can also be used to estimate the sensitivity of each RBF to a random displacement. If we move an RBF having a high singular value, we will probably obtain a worse-fitted net. On the other hand, RBFs whose singular values are nearly zero are not making any significant contribution to the net output, so they can be altered freely without increasing the net error.

With this idea in mind, this mutation operator selects a basis function to be altered with a probability that is inversely proportional to its associated singular value. Less relevant RBFs will have more probability of being altered while more important RBFs will be more change-protected [35]. Once an RBF has been chosen, a local modification is applied to its center or radius as described in Sections IV-E–1 and IV-E-2.

G. LOLSM: Local OLS-Based Mutation

OLS calculates a vector of error reduction ratios $\boldsymbol{err}$ (11) in which there is an $[\mathrm{err}]_j$ for each RBF ϕ_j in the net. The higher $[\mathrm{err}]_j$ is, the more sensitive ϕ_j is to a random change. Thus, this mutation operator constructs a probability distribution where each ϕ_j has a probability of being altered that is inversely proportional to its associated error reduction ratio $[\mathrm{err}]_j$ [35]. An

RBF is chosen using this distribution and a local change is applied to its center or radius as described in Sections IV-E-l and IV-E–2.

H. GSVDM: Global SVD-Based Mutation

Another way of hybridization between SVD and a mutation operator is to select the RBF to be altered uniformly (all the RBFs have the same likelihood of being chosen) and apply a random displacement according to its associated singular value. Basis functions with small singular values will undergo large movements, while only small perturbations will be applied to sensitive RBFs.

This behavior can be implemented by applying a random shift to the center or the radius of the selected RBF whose modulus varies inversely with the magnitude of its singular value. The steps to perform the alterations are detailed later (Sections IV-H-1 and IV-H-2).

1) Globally SVD-Based Random Change of the Center of an RBF: To alter the position of an RBF globally, a random displacement is applied to its center. The great difference from the mutation operators described earlier is that in this case, the RBF movement can be made toward any point in the whole input space. The modulus of the displacement applied increases as the RBF becomes less important, allowing large movements to basis functions that contribute insignificantly to the net output. Relevant RBFs will only undergo small alterations, to avoid a large change in net performance.

Each one of the offset components is calculated as follows:

$$\mathrm{off}_i = \mathrm{sign}() \cdot \mathrm{u}(0, r^j + \delta^j) \tag{32}$$

where $\mathrm{sign}()$ is the function described in (23), r^j is the radius of the selected basis function ϕ_j, and δ^j is a constant that is inversely proportional to the singular value associated with ϕ_j

$$\delta^j = \frac{\sigma_{\max} - \sigma_j}{\sigma_{\max} - \sigma_{\min}} \cdot \Delta^j. \tag{33}$$

In the above equation, $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and minimum singular values, respectively, σ_j is the singular value corresponding to ϕ_j, and Δ^j is the maximum allowed movement for the jth basis function

$$\Delta^j = \max\left(\max_i\left\{\left|M_i - c_i^j\right|\right\}, \max_i\left\{\left|c_i^j - m_i\right|\right\}\right) \quad 1 \le i \le d \tag{34}$$

M_i and m_i are defined in (26) and (27) respectively, and d is the input space dimensionality.

Moreover, this random displacement must satisfy constraints (24) and (25) to ensure that the altered RBF will be activated by some training data.

2) Globally SVD-Based Random Change of the Radius of an RBF: This alteration is also performed by a random shift applied to the RBF radius. The change of the radius is described as

$$(r^j)' = r^j + \mathrm{sign}() \cdot \mathrm{u}(0, r^j + \delta^j) \tag{35}$$

where δ^j is calculated from (33).

Once the alteration has been performed, the new RBFNN must satisfy constraints (29), (30) and (31) to be accepted.

I. GOLSM: Global OLS-Based Mutation

This evolutionary operator implements exactly the same steps as above, but uses OLS instead of SVD to detect the relevance of the RBFs. All the basis functions have the same likelihood of being altered, and when one RBF ϕ_j is chosen, its center or its radius undergoes a random displacement that is inversely proportional to its error reduction ratio $[\mathrm{err}]_j$ (11).

1) Globally OLS-Based Random Change of the Center of an RBF: This alteration is analogous to the one performed by GSVDM. The only difference is the calculation of the constant δ^j, which is now based on the error reduction ratios

$$\delta^j = \frac{[\mathrm{err}]_{\max} - [\mathrm{err}]_j}{[\mathrm{err}]_{\max} - [\mathrm{err}]_{\min}} \cdot \Delta^j \tag{36}$$

where $[\mathrm{err}]_{\max}$ and $[\mathrm{err}_{\min}$ are the maximum and minimum error reduction ratios obtained by the OLS decomposition respectively, $[\mathrm{err}]_j$ is the error reduction ratio of the chosen RBF, and Δ^j is obtained by applying (34).

Again, the random displacement applied to the RBF must meet restrictions (24) and (25) to produce a valid RBFNN.

2) Globally OLS-Based Random Change of the Radius of an RBF: This alteration also performs a random displacement of the current RBF radius. This change is applied according to

$$(r^j)' = r^j + \mathrm{sign}() \cdot \mathrm{u}(0, r^j + \delta^j) \tag{37}$$

where δ^j is obtained with (36).

This random alteration must also satisfy constraints (29), (30) and (31) to be validated.

V. Comparison of the Parameter Adjustment Mutation Operators

As commented in Section II, SVD and OLS have been widely applied as heuristics to prune the less relevant units of an RBFNN [14]–[16], [45], [58], [85], [86]. Nevertheless, this paper also uses them to guide the adjustment of the net parameters. The effect of this new use of these heuristics has not yet been analyzed, and so this section presents an experiment to gain an insight into the behavior of the adjustment of the parameters for each mutation operator. This experiment is related to the approximation of the target function

$$f(x) = 3x(x-1)(x-1.9)(x+0.7)(x+1.8), \quad x \in [-2.1, 2.1] \tag{38}$$

proposed in [23] from a set of 100 equi-distributed samples. This function was approximated with several EAs, each one containing the crossover operator and only one of the parameter adjustment mutation operators presented earlier (LRM, LSVDM,

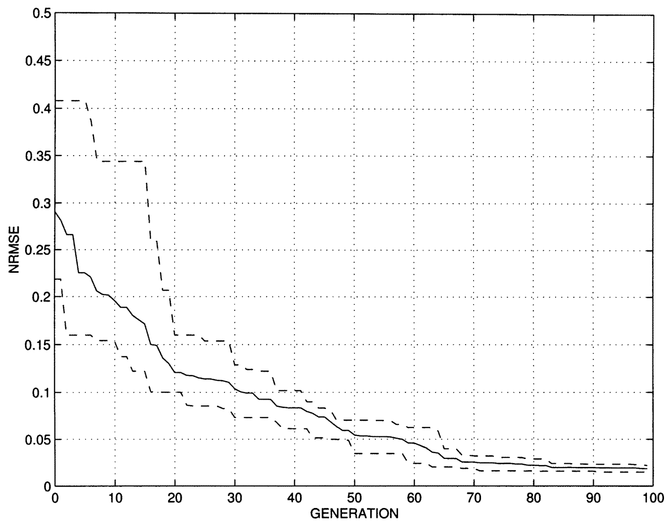

Fig. 3. Effect of LRM operator.

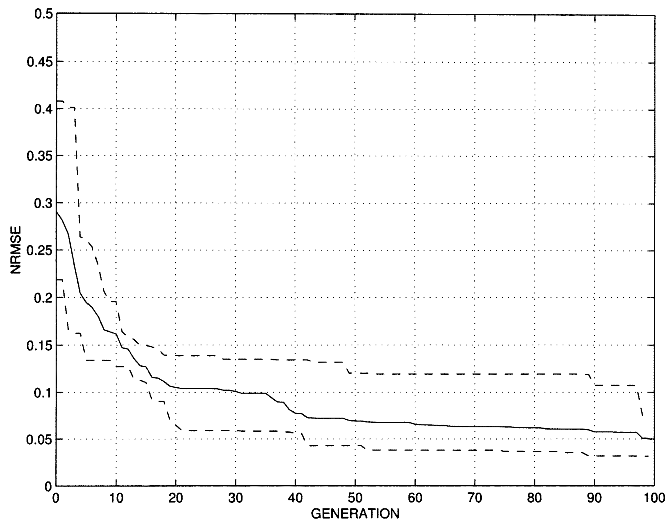

Fig. 4. Effect of LOLSM operator.

LOLSM, GSVDM, GOLSM)[2]. These EAs were used with different initial populations and all the executions were started with the same random seed. All the EAs used a population composed of 30 individuals each with 7 RBFs, and all of them were run for 100 generations. These parameters were fixed arbitrarily, but as they are identical for all the EAs, they provide a fair comparison between the mutation operators.

The index used to estimate the approximation error of the RBFNNs is the normalized root mean squared error (NRMSE) [65], defined as

$$\mathrm{NRMSE} = \sqrt{\frac{\sum_{k=1}^{n}(y^k - \mathcal{F}(x^k;\Phi,\Omega))^2}{\sum_{k=1}^{n}(y^k - \bar{y})^2}} \tag{39}$$

where y^k are the expected outputs for the n training examples and $\bar{y}$ represents their mean.

Figs. 3–7 show the evolution of the different EAs. The solid line represents the mean of the best individuals in each generation, and the dashed line indicates the minima and maxima of the best individuals found in every generation. These figures suggest that LRM achieves a very good solution, even better than LOLSM and LSVDM. *This detail reveals that even with expert knowledge the search can be trapped in local minima if it is not correctly used.* As LOLSM and LSVDM only perform local changes to the individuals, they prevent less relevant RBFs from moving freely in the input space. Although less important RBFs are given more chances of being altered, as the random changes are always made in a local way, they only affect contiguous regions in the input space. These local alterations do not allow an RBF which makes little contribution to the net output to move to a region in the input space where it could reduce the approximation error, if this region is not close to the RBF. Thus, these two mutation operators tend to trap the population in local minima, and *this is just the opposite function a mutation operation should perform.*

On the other hand, the GOLSM and GSVDM operators obtain much better results. This is because GOLSM and GSVDM

[2]Note that as in this section we only intend to analyze the behavior of the parameter adjustment mutation operators, we only use an EA with a fixed number of RBFs for all the RBFNNs, not the MOEA described in Section III.

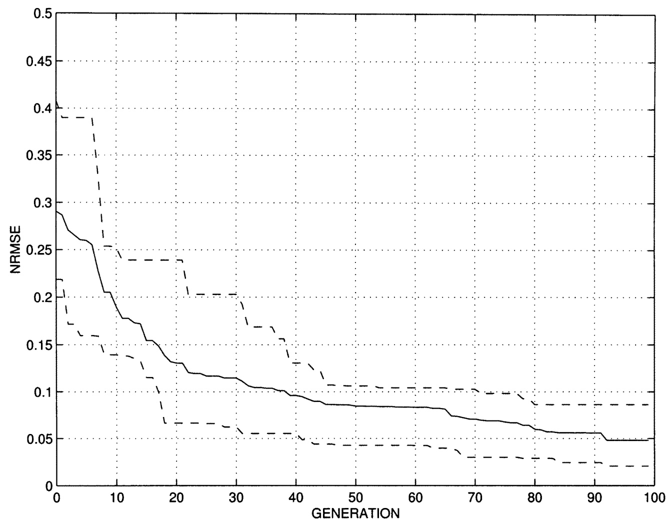

Fig. 5. Effect of LSVDM operator.

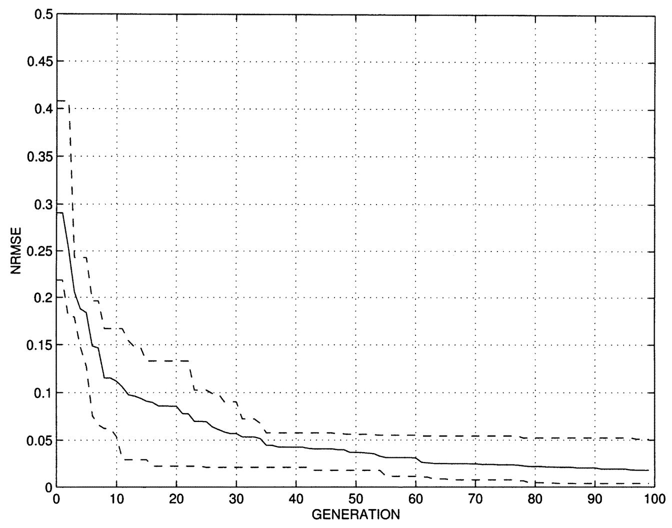

Fig. 6. Effect of GOLSM operator.

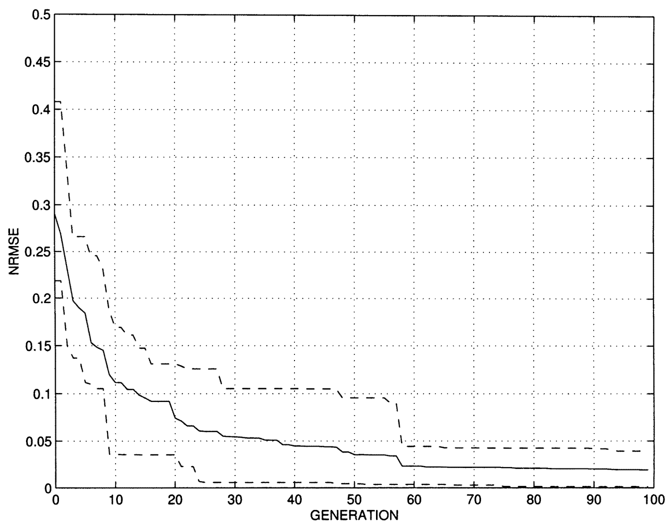

Fig. 7. Effect of GSVDM operator.

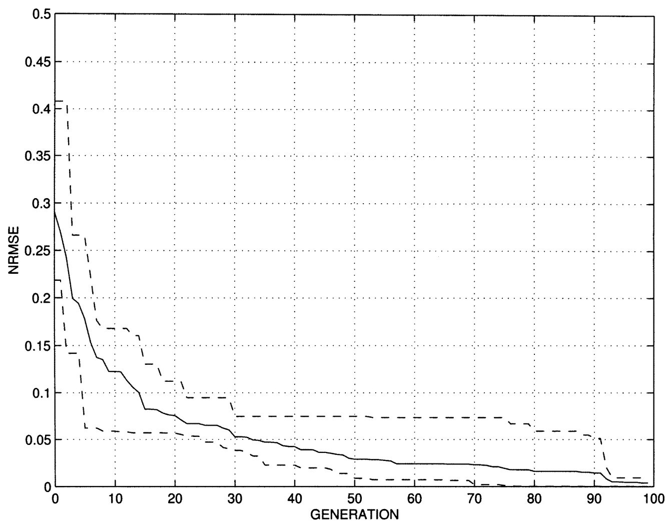

Fig. 8. Combined effect of GOLSM and GSVDM operators.

TABLE II
MEAN, STANDARD DEVIATION, MINIMUM, AND MAXIMUM OF THE BEST SOLUTIONS FOUND FOR EACH MUTATION OPERATOR

Operator	Mean	St. Dev.	Minimum	Maximum
LRM	0.0186	0.0022	0.0150	0.0219
LOLSM	0.0507	0.0159	0.0316	0.0753
LSVDM	0.0481	0.0253	0.0208	0.0864
GOLSM	0.0178	0.0173	0.0039	0.0510
GSVDM	0.0192	0.0141	0.0013	0.0392
GOLSM + GSVDM	0.0039	0.0031	0.0002	0.0093

perform global changes to the individuals using displacements that are inversely proportional to the relevance of the RBF they affect. Thus, important basis functions will only undergo local perturbations and less relevant RBFs will be able to move freely.

Figs. 6 and 7 show that GOLSM and GSVDM require less iterations than LRM to reach EA convergence. Starting with the same initial populations, GOLSM and GSVDM are able to discover solutions with an approximation error that is half that of the EA using LRM in the same generation. This result justifies the selection of GOLSM and GSVDM as the most appropriate mutation operators for this problem.

If we now pay attention to the last generations, LRM, GOLSM and GSVDM produce similar mean approximation errors, although the best solution is always better for GOLSM and GSVDM. This comparison is graphically shown in Fig. 9 and Table II.

Finally, as we can add as many mutation operators as we want to an EA, the same experiments were run with an EA combining GOLSM and GSVDM. When an individual is chosen to be altered, one of these two operators is applied. The application probability was 0.5 for each of them. This final EA shows an easy way of combining the effects of several operators. Figs. 8–9 and Table II show that this combination produces even better solutions than when each operator is applied separately.

As a conclusion, the idea of a local mutation operator is a contradiction, given that applying local changes tend to trap the population in local minima, instead of facilitating a way of escaping from them. Nevertheless, global mutation operators can move less relevant (less activated) RBFs through the whole input space, facilitating the allocation of these useless RBFs in other positions where they might become more activated and perform a higher contribution to the approximation error reduction. This conclusion is in line with the state-of-the-art quantization algorithm known as ELBG [63], where local minima in the parameter space are avoided throughout a genetic-based global migration procedure to allow codewords to move through noncontiguous Voronoi polyhedra.

VI. EXPERIMENTAL RESULTS

Having analyzed the parameter adjustment mutation operators, this section shows some experimental results obtained by the MOEA described in Section III incorporating the crossover operator, the pruning operators SVDP, OLSP, the RBF splitting operator SPLIT, and the parameter adjustment mutation operators GOLSM and GSVDM. The proposed multiobjective evolutionary algorithm has been checked in the fields of function approximation and chaotic time series.

As described earlier, the proposed algorithm is able to obtain in only one execution several optimum solutions for different configurations (a Pareto-optimum frontier of solutions) for a given training set of examples. Thus, in the tables that summarize the experimental results will be presented RBFNNs with several complexities together with their approximation error.

A. Application to Function Approximation

The first experiment tests the MOEA proposed in this paper to approximate several one-dimensional (1-D) and two-dimensional (2-D) target functions proposed in the literature.

1) 1-D Functions: In this section the proposed algorithm is tested with three 1-D functions previously used by other authors. The results obtained are compared with the solutions presented by other authors in terms of the error committed by the model and its complexity.

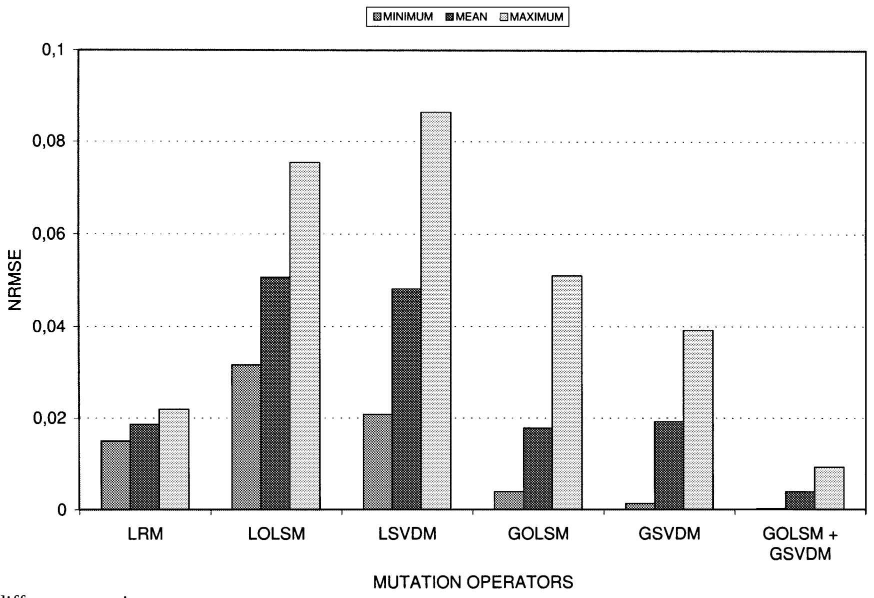

Fig. 9. Comparison of the different mutation operators.

TABLE III
COMPARISON OF THE OBTAINED RESULT WITH OTHER APPROACHES USED TO APPROXIMATE THE 1-D TARGET FUNCTION dick; m REPRESENTS THE NUMBER OF RBFs OR RULES (DEPENDING ON THE MODEL), AND n_p IS THE NUMBER OF FREE PARAMETERS

Algorithm			m	n_p	MSE	NRMSE
Dickerson & Kosco, 1996	Different weights for the rules	$\omega_k = 1$	6	–	94.65	–
		$\omega_k = 1/v_k$			28.25	–
		$\omega_k = 1/v_k^2$			10.53	–
	Not supervised				7.927	–
	Supervised				3.069	–
Pomares			5	8	5.01	0.329
			6	10	1.35	0.17
			7	12	0.46	0.10
Proposed approach			3	9	5.57 ± 0	0.3455 ± 0
			4	12	0.99 ± 0.49	0.1415 ± 0.0390
			5	15	0.30 ± 0.02	0.0797 ± 0.0023

The first 1-D target function used in this section was originally proposed by Dickerson and Kosko in [23]. It has been previously used in Section V to test the behavior of the mutation operators and in this section it will be used as a test function to compare the proposed algorithm with other models and algorithms proposed in the literature for function approximation. The function is defined as

$$\mathrm{dick}(x) = 3x(x-1)(x-1.9)(x+0.7)(x+1.8), \quad x \in [-2.1, 2.1]. \quad (40)$$

The MOEA was run several times with different populations of 25 RBFNNs. The training set used was composed of 100 examples equidistributed in the input interval $[-2.1, 2.1]$ and the test set contained 1000 test data also equidistributed in the same input range.

Table III shows the approximation error reached by the algorithms proposed in [23], [65]. Dickerson and Kosko applied a hybrid neuro-fuzzy system with ellipsoidal rules trained by several learning methods, while Pomares proposed a fuzzy system based on a complete table of rules using triangular membership functions. The error is compared using two different indexes: the MSE proposed originally in [23], and the NRMSE proposed in [65]. It can be observed that the standard deviations over the mean approximation error are quite low for the different structures found by the proposed algorithm, which reveals the robustness of the proposed algorithm for different random initial populations.

The other two 1-D target functions used to test the proposed algorithm were proposed in [80], one of the first works that addressed the problem of function approximation from a set of training examples. This algorithm generated a fuzzy rule-table having one rule for each training example and later selected the more activated rules to construct the model. Later on, Sudkamp and Hamell [78] improved this algorithm to make it noise resis-

TABLE IV
COMPARISON OF THE OBTAINED RESULT WITH OTHER APPROACHES USED TO APPROXIMATE THE 1-D TARGET FUNCTION $wm_I(x)$; m REPRESENTS THE NUMBER OF RBFS OR RULES (DEPENDING ON THE MODEL), AND n_p IS THE NUMBER OF FREE PARAMETERS

Algorithm		m	n_p	Mean Err.	NRMSE
Wang & Mendel, 1992		15	–	0.029	–
		25	–	0.018	–
Wang & Mendel Improvement (in Sudkamp & Hamell, 1994)		15	–	0.079	–
		25	–	0.088	–
Sudkamp & Hamell, 1994	Region Growing	15	–	0.044	–
		25	–	0.021	–
	Weighted Average	15	–	0.044	–
		25	–	0.021	–
Pomares, 2000		4	6	0.026	0.080
		6	10	0.011	0.032
		8	14	0.006	0.017
Proposed Algorithm		2	6	0.0057 ± 0.0007	0.0171 ± 0.0018
		3	9	0.0042 ± 0.0005	0.0134 ± 0.0011
		4	12	0.0001 ± 5.4E-5	0.0004 ± 0.0002
		5	15	2.1E-5 ± 1.8E-5	6.8E-5 ± 5.7E-5

tant and also proposed other two methods: *Region Growing* and *Weighted Average*. To test the algorithms both works used the target functions

$$wm_I(x) = x^3, \quad x \in [-1, 1] \tag{41}$$

$$wm_{II}(z) = \sin(2\pi x), \quad x \in [-1, 1]. \tag{42}$$

These two functions were also used by Pomares to test its systematic learning algorithm for the identification of rule-based fuzzy systems in [65].

The algorithm proposed in this paper was run several times with different populations of 25 RBFNNs. The training set used to learn these two functions were constructed with 100 examples equidistributed in their input range, while the test sets was formed by 1000 examples uniformly distributed in the same interval.

Tables IV and V compare the results of the proposed algorithm with the obtained by the other approaches introduced earlier. In this case, the error index used in [80], [78] was the mean error (mean of the absolute errors). Pomares also used in [65] the NRMSE, so the results are compared taking into account the two indexes. As can be seen, the proposed algorithm obtains approximations with a lower approximation error, even using fewer free parameters. The standard deviation of the approximations obtained after several runs also show that the algorithm is quite robust. This can be deduced because it is able to reach similar solutions starting with different random initial populations.

Fig. 10 and 11 rank the solutions obtained by all the algorithms compared in this section according to the two objectives being minimized: the approximation error and the complexity of the model[3]. It can be appreciated that the proposed algorithm is able to explore the Pareto-optimum frontier, finding solutions that are superior in both competing objectives.

TABLE V
COMPARISON OF THE OBTAINED RESULT WITH OTHER APPROACHES USED TO APPROXIMATE THE 1-D TARGET FUNCTION $wm_{II}(x)$; m REPRESENTS THE NUMBER OF RBFS OR RULES (DEPENDING ON THE MODEL), AND n_p IS THE NUMBER OF FREE PARAMETERS

Algorithm		m	n_p	Mean Err.	NRMSE
Wang & Mendel, 1992		15	–	0.060	–
		25	–	0.026	–
Wang & Mendel Improvement (in Sudkamp & Hamell, 1994)		15	–	0.071	–
		25	–	0.031	–
Sudkamp & Hamell, 1994	Region Growing	15	–	0.131	–
		25	–	0.052	–
	Weighted Average	15	–	0.180	–
		25	–	0.082	–
Pomares, 2000		6	10	0.068	0.112
		8	14	0.0473	0.0823
		10	18	0.0262	0.0237
Proposed algorithm		3	9	0.0582 ± 0.0003	0.1169 ± 4.0E-5
		4	12	0.0107 ± 0.0111	0.0200 ± 0.0237
		5	15	0.0068 ± 0.0068	0.0143 ± 0.0173
		6	18	0.0028 ± 0.0013	0.0049 ± 0.0023

2) 2-D Functions: In this section, we have used the 2-D functions f_5 and f_7 (Figs. 12 and 13) originally proposed in [18]. This work presented a comparative study of several paradigms applied to function approximation, such as Projection Pursuit (PP) [28], Multivariate Adaptive Regression

[3]The complexity is measured using the number of rules (or RBFs in an RBFNN).

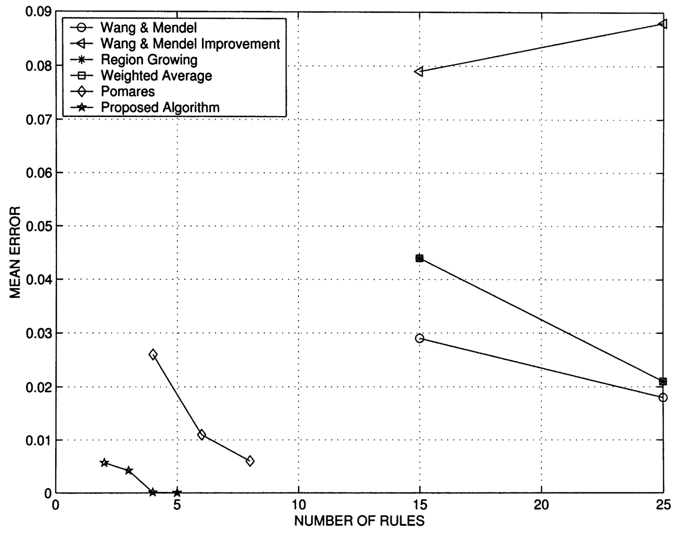

Fig. 10. Comparison of the proposed algorithm with others applied in the literature to approximate thee target function $wm_I(x)$.

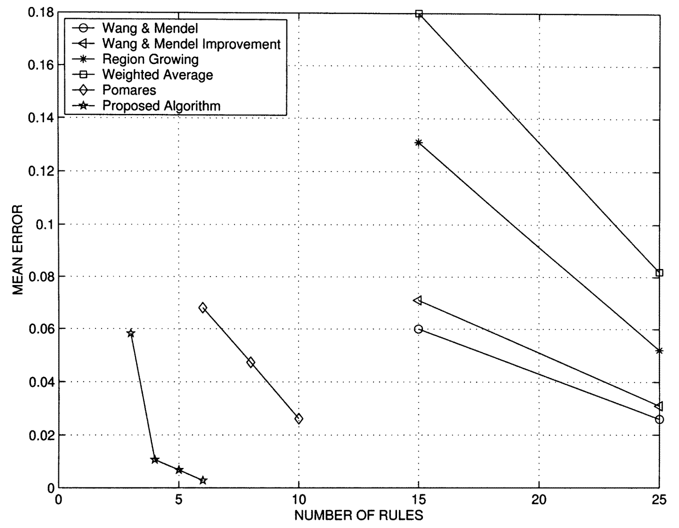

Fig. 11. Comparison of the proposed algorithm with others applied in the literature to approximate thee target function $wm_{II}(x)$.

Splines (MARS) [29], Constrained Topological Mapping (CTM), and a multilayer perceptron (MLP) with 15 neurons in the hidden layer. These two target functions are defined as follows:

$$f_5(x_1, x_2) = 42.659\left(0.1 + x_1\left(0.05 + x_1^4 - 10x_1^2x_2^2 + 5x_2^4\right)\right) \quad x_1, x_2 \in [-0.5, 0.5] \quad (43)$$

$$f_7(x_1, x_2) = 1.9(1.35 + e^{x_1}\sin(13(x_1 - 0.6)^2)e^{-x_2}\sin(7x_2)) \quad x_1, x_2 \in [0, 1]. \quad (44)$$

Later on, this functions were also used in [17], a paper presenting a very optimized method to construct MLPs is presented, and in [65], where a robust algorithm for the identification of rule-based fuzzy systems is used in function approximation problems. This algorithm was able to use different types of membership functions, such as a triangular partition (TP) of the input space, free triangular (FT) membership functions in the input space, Gaussian functions (G), and free pseudo-Gaussian (FPG) functions. The target function

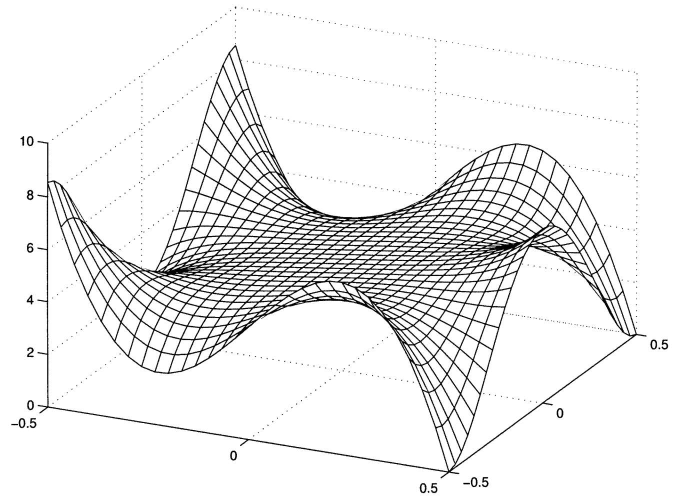

Fig. 12. Function f_5.

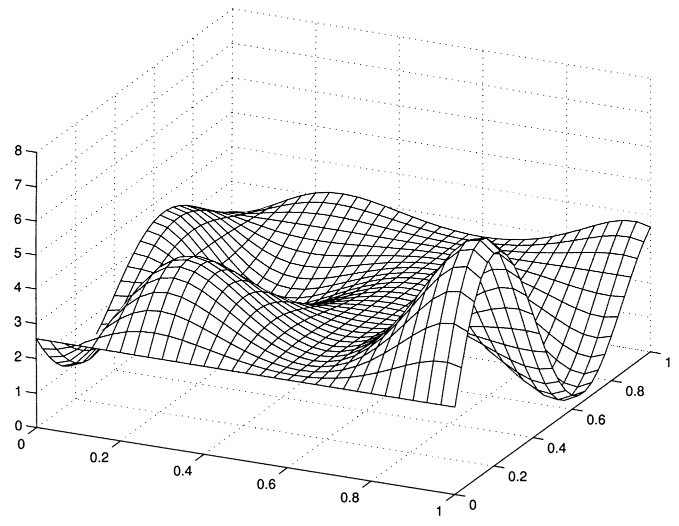

Fig. 13. Function f_7.

f_7 was also used as a test function in [10], [11]. These works describe the G-Prop algorithm, an evolutionary algorithm for multilayer perceptrons.

The training sets for the experiments presented in these sections have been constructed taking a random point from each cell of a 20 × 20 grid partition of the input space, obtaining similar training sets to the ones used in [17]. The test sets were formed by 961 points obtained by dividing the input interval with a (31 × 31) grid.

As can be appreciated in Tables VI and VII, the models obtained for each target function are superior to all classic approaches (MLP, PP, CTM y MARS), as they obtain quite lower approximation errors. The MLPs obtained by Cherkassky *et al.* in [17] and by Castillo in [10], [11] are also outperformed in approximation error and in the complexity of the models in the approximation of f_7.

The fuzzy systems obtained in [65] present a very low approximation error due to the great number of linear parameters that can be optimally calculated in rule-based fuzzy systems. This fact becomes maximized when the membership functions are a triangular partition of the input space. Due to this characteristic, the results obtained by the proposed approach are similar

TABLE VI
COMPARISON OF THE OBTAINED RESULT WITH OTHER APPROACHES USED TO APPROXIMATE THE 2-D TARGET FUNCTION f_5, m REPRESENTS THE NUMBER OF RBFs OR RULES (DEPENDING ON THE MODEL), AND n_p IS THE NUMBER OF FREE PARAMETERS

Algorithm	m	n_p	Test NRMSE
MLP (Cherkassky, 1991)	15	61	0.308
PP (Friedman, 1981)	–	–	0.504
CTM (Cherkassky, 1991)	–	–	0.131
MARS (Friedman, 1991)	–	–	0.190
Cherkassky *et al*, 1996	40	161	0.038
Pomares, 2000	4×5 (PT)	25	0.194
	6×6 (PT)	44	0.090
	8×8 (PT)	76	0.044
Proposed Algorithm	6	24	0.2864 ± 4.6E-5
	7	28	0.1705 ± 0.1237
	8	32	0.1170 ± 0.0852
	9	36	0.0738 ± 0.0681
	10	40	0.0689 ± 0.0517
	11	44	0.0422 ± 0.0207
	12	48	0.0285 ± 0.0109
	13	52	0.0235 ± 0.0008
	14	56	0.0216 ± 0.0055
	15	60	0.0175 ± 0.0059
	16	64	0.0168 ± 0.0096
	17	68	0.0154 ± 8.5E-5
	18	72	0.0138 ± 0.0054
	19	76	0.0121 ± 0.0038

TABLE VII
COMPARISON OF THE OBTAINED RESULT WITH OTHER APPROACHES USED TO APPROXIMATE THE 2-D TARGET FUNCTION f_7; m REPRESENTS THE NUMBER OF RBFs OR RULES (DEPENDING ON THE MODEL), AND n_p IS THE NUMBER OF FREE PARAMETERS

Algorithm	m	n_p	Test NRMSE
MLP (Cherkassky, 1991)	15	61	0.227
PP (Friedman, 1981)	–	–	0.206
CTM (Cherkassky, 1991)	–	–	0.197
MARS (Friedman, 1991)	–	–	0.179
Cherkassky *et al*, 1996	40	161	0.034
Pomares, 2000	4×4 (PT)	20	0.161
	5×5 (PT)	31	0.109
	7×6 (PT)	51	0.059
Castillo, 2000	G-Prop (fn)	118 ± 39	0.21 ± 0.01
	G-Prop (fl)	105 ± 34	0.23 ± 0.01
	G-Prop (fd)	115 ± 36	0.22 ± 0.02
Proposed Algorithm	4	16	0.3433 ± 0.0002
	5	20	0.2639 ± 0.0002
	6	24	0.2019 ± 0.0243
	7	28	0.1426 ± 0.0190
	8	32	0.1306 ± 0.0189
	9	36	0.1086 ± 0.0316
	10	40	0.0780 ± 0.0080
	11	44	0.0711 ± 0.0093
	12	48	0.0608 ± 0.0165
	13	52	0.0590 ± 0.0103
	14	56	0.0557 ± 0.0141
	15	60	0.0493 ± 0.0057
	16	64	0.0474 ± 0.0062
	17	68	0.0443 ± 0.0032
	18	72	0.0393 ± 0.0024
	19	76	0.0324 ± 0.0050

to those presented in [65] for function f_7. Nevertheless, function f_5 has been learned better for the proposed algorithm, as can be seen in Fig. 14.

3) Robustness to Noise in the Training Data: The experiments above have been performed with ideal training sets in order to compare the proposed algorithm to others in the literature. Nevertheless, in real experiments, data usually are affected some noise. To give an insight of the proposed algorithm behavior modeling noisy data, in this experiment a 5% of white noise has been added to the training data used before to learn the target function f_5. The effect of this noise can be observed in Fig. 15.

The proposed algorithm has been trained several times with the noisy training data and with the same configuration used in Section VI-A–2. The obtained results, shown in Table VIII, show the robustness of the proposed algorithm facing noisy data. As the number of RBFs increases, the data are better modeled and the test error diminishes, until a sufficient number of RBFs is reached, nine in this case (see Fig. 16). RBFNNs having more than nine RBFs also learn some of the noise added to the training data, thus the test error increases. This threshold for the maximum number of RBFs allowed can be easily found when detecting more complex RBFNNs with higher test approximation errors.

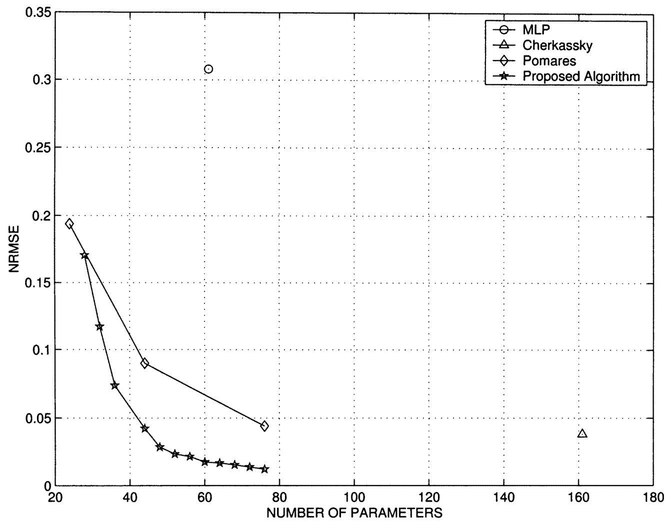

Fig. 14. Comparison of the proposed algorithm with others applied in the literature to approximate thee target function f_5.

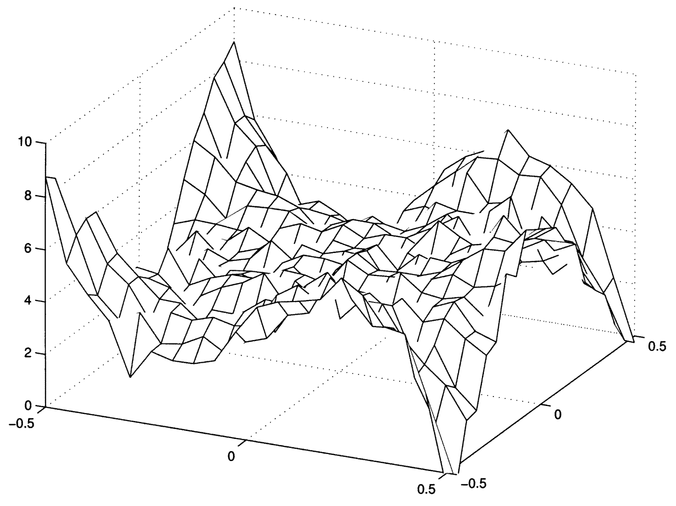

Fig. 15. Noisy training data used to learn the target function f_5.

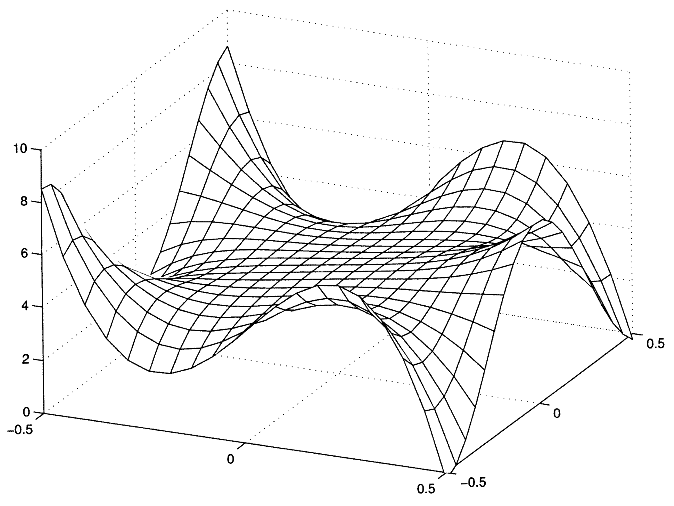

Fig. 16. Approximation of function f_5 with nine RBFs after training with noisy data.

TABLE VIII
TRAINING AND TEST APPROXIMATION ERRORS OBTAINED USING A NOISY TRAINING SET TO LEARN THE f_5

Num.	Training NRMSE			Test NRMSE		
RBFs	Min.	Mean	St. Dev.	Min.	Mean	St. Dev.
2	0.9118	0.9118	0	0.8922	0.8922	0
3	0.7385	0.7385	0	0.7014	0.7014	0
4	0.6423	0.6423	0	0.6013	0.6013	0
5	0.5435	0.5518	0.0099	0.4967	0.5027	0.0072
6	0.4137	0.4137	0	0.3044	0.3044	0
7	0.3237	0.3740	0.0294	0.0985	0.2401	0.0795
8	0.3200	0.3405	0.0277	0.1054	0.1763	0.0619
9	0.3205	0.3256	0.0060	0.1047	0.1410	0.0357
10	0.3123	0.3208	0.0065	0.1186	0.1852	0.0955
11	0.3145	0.3167	0.0030	0.1304	0.1902	0.0445
12	0.3117	0.3140	0.0022	0.1314	0.7012	1.0595
13	0.3085	0.3127	0.0038	0.1332	0.2361	0.0897
14	0.3026	0.3081	0.0042	0.1498	0.4082	0.2698
15	0.2992	0.3065	0.0051	0.1566	0.2333	0.0990
16	0.2982	0.3042	0.0053	0.1738	0.4449	0.4624
17	0.3020	0.3034	0.0020	0.1586	0.2493	0.0732
18	0.2942	0.3010	0.0050	0.2202	0.4500	0.3662
19	0.2967	0.2992	0.0022	0.4745	0.8283	0.4047

B. Application to Time Series Forecasting

The MOEA proposed in this paper has also been tested with the time series generated by the Mackey-Glass time-delay differential equation [53]

$$\frac{ds(t)}{dt} = \alpha \cdot \frac{s(t-\tau)}{1+s^{10}(t-\tau)} - \beta s(t). \qquad (45)$$

Following previous studies [84], the parameters were fixed to $\alpha = 0.2, \beta = 0.1$, thus obtaining a chaotic time series with no clearly defined period; it does not converge or diverge, and is very sensitive to initial conditions.

As in [43], the time series values at integer points were obtained applying the fourth-order Runge-Kutta method to find the numerical solution for the above equation. The values $s(0) = 1.2, \tau = 17$, and $s(t) = 0$ for $t < 0$ were assumed. This data set can be found in the file mgdata.dat belonging to the FUZZY LOGIC TOOLBOX OF MATLAB 5.

1) Short-Term Prediction: Following the conventional settings to perform a short-term prediction of these time series, we predict the value $s(t+6)$ from the current value $s(t)$ and the past values $s(t-6), s(t-12)$, and $s(t-18)$; thus, the training vectors for the model have the following format

$$[s(t-18), s(t-12), s(t-6), s(t), s(t+6)]. \qquad (46)$$

The first 500 input vectors were used to train the model and the next 500 vectors were used to test the RBFNNs obtained. The proposed algorithm was run several times with a population of 25 individuals for 1000 generations, and the Levenberg-Marquardt minimization algorithm was applied to the best solutions found to fine-tune their parameters. Table IX compares the obtained result with other presented in the literature in terms of their root mean squared error (RMSE), defined as

$$\text{RMSE} = \sqrt{\frac{\sum_{k=1}^{n}(y^k - \mathcal{F}(x^k; \Phi, \Omega))^2}{n}} \qquad (47)$$

The comparison of these results with those obtained by the proposed algorithm reveals that the use of expert mutation operators significantly enhances the search results. This can be observed in the standard deviations from the mean RMSE, which are small enough to establish the robustness of our approach. Fig. 17 shows the Pareto-optimum solutions found by the proposed algorithm and compares them with other solutions in the literature. It can be observed that the proposed approach is able to find a wide range of solutions with different compromises between the number of RBFs and the approximation error, and that all the solutions are found in only one execution of the algorithm.

2) Long-Term Prediction: Although the short-term prediction of the Mackey-Glass time series has been used as a test benchmark by several classical methods, as discussed in [52],

TABLE IX
COMPARISON OF THE PROPOSED ALGORITHM WITH OTHERS APPLIED IN THE LITERATURE TO PREDICT THE $s(t+6)$ VALUE OF THE MACKEY-GLASS TIME SERIES; m REPRESENTS THE NUMBER OF RBFS OR RULES (DEPENDING ON THE MODEL), AND n_p IS THE NUMBER OF FREE PARAMETERS

Algorithm		m	n_p	Test RMSE
Linear Predictive Method		–	–	0.55
Auto-Regresive Model		–	–	0.19
L-X. Wang	T-Norm: Prod.	–	–	0.0907
	T-Norm: Min.	–	–	0.0904
Cascade Correlation ANN		–	–	0.06
6th Order Polynomial		–	–	0.04
D. Kim y C. Kim (Genetic Algorithm + Fuzzy System)		$5\times5\times5\times5$ (TL)	665	0.0492
		$7\times7\times7\times7$ (TL)	2457	0.0423
		$9\times9\times9\times9$ (TL)	6633	0.0379
Retropropagation ANN		–	–	0.02
ANFIS (ANN + Fuzzy Logic)		–	–	0.007
Pomares, 2000		$2\times3\times3\times3$ (PT)	57	0.0109
		$3\times4\times4\times4$ (PT)	199	0.0071
		$4\times4\times5\times5$ (PT)	410	0.0063
		$2\times2\times2\times2$ (PGL)	24	0.0203
		$3\times3\times3\times3$ (PGL)	101	0.0058
Proposed Approach		4	24	0.0151 ± 0.0019
		5	30	0.0120 ± 0.0026
		6	36	0.0090 ± 0.0014
		7	42	0.0075 ± 0.0009
		8	48	0.0064 ± 0.0012
		9	54	0.0060 ± 0.0010
		10	60	0.0055 ± 0.0010
		11	66	0.0049 ± 0.0010
		12	72	0.0047 ± 0.0009
		13	78	0.0043 ± 0.0011
		14	84	0.0045 ± 0.0007
		15	90	0.0039 ± 0.0002
		16	96	0.0036 ± 0.0006
		17	102	0.0034 ± 0.0003
		18	108	0.0032 ± 0.0007
		19	114	0.0030 ± 0.0006
		20	120	0.0029 ± 0.0006

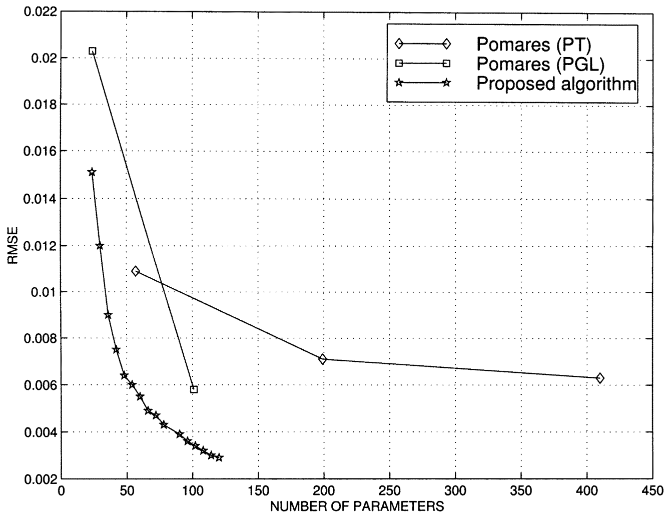

Fig. 17. Comparison of the proposed algorithm with others applied in the literature to predict the $s(t+6)$ value of the Mackey-Glass time series.

[57], the prediction lead time (the number of steps in the future to be predicted) should be greater than the characteristic time of this chaotic time series. They and most of the authors who have this time series as a serious benchmark predict at least 85 time steps into the future, i.e., they predict the value of time series at time $t+85$ from the four values a times $t, t-6, i-12$, and $i-18$. As argued by Moody and Darken in [57], this prediction problem is a significant challenge in which classical methods do little better than chance, thus the use of RBFNNs is justified.

As in the case of short-term prediction, the first 500 vectors have been used in the training step and the remaining 500 have been used o validate the RBFNNs returned by the MOEA. Table X compares the results returned by the proposes algorithm with several approaches used to solve this problem in the literature. Some of them are also based on RBFNNs, such as the model RAN [64], which iteratively constructs an RBFNN analyzing the novelty of the input data, or the modifications of RAN proposed in [72], which include the Givens QR decomposition (RAN-GQRD) to obtain the weights of the net and a pruning criterion (RAN-P-GQRD) to reduce the complexity of the net. The results are compared with other paradigms too. One of them [3] presents two different algorithms to train fuzzy systems, one using brute force and another incremental, and it is shown that the brute force approach presents an unstable behavior as the number of rules is increased and it not reaches the approximation errors obtained by the incremental algorithm. The other one [20] applies EGAs *(Breeder Genetic Algorithms)* to train MLPs. Again, it can be appreciated that the proposed algorithm is able to find a set of Pareto-optimum solutions that dominate all the solutions in the table. Fig. 18 summarizes graphically the results.

VII. CONCLUSION

This paper presents a set of new mutation operators specially designed to evolve RBFNNs. These operators incorporate expert knowledge of the problem in order to favor random changes that may improve the affected individuals instead of performing only blind changes. The orthogonal transformations OLS and SVD were applied to the activation matrix of the model. As these transformations provide a way of ranking the RBFs according to their relevance in the net, they are used by the mutation operators to decide which hidden unit should be modified to improve the net approximation error.

Nevertheless, *the incorporation of expert knowledge and heuristics into the mutation operators does not always produce a good EA.* This paper has shown two clear examples (LOLSM and LSVDM) where expert mutation operators do not favor the convergence to good local optima. The objective of mutation operators is to add diversity to the population and to provide mechanisms to favor the exploration of the search space. If the expert knowledge limits this objective in any way, the EA will not search properly and will tend to get stuck in a local optimum. On the other hand, if the heuristics cleverly guide the alterations toward better solutions, an EA using such operators can achieve better solutions than a blind one. This is illustrated

TABLE X
COMPARISON OF THE PROPOSED ALGORITHM WITH OTHERS APPLIED IN THE LITERATURE TO PREDICT THE $s(t+85)$ VALUE OF THE MACKEY-GLASS TIME SERIES; m REPRESENTS THE NUMBER OF RBFs OR RULES (DEPENDING ON THE MODEL), AND n_p IS THE NUMBER OF FREE PARAMETERS

Algorithm		m	n_p	Test NRMSE
MLP + BGA (De Falco *et al.* 1998)		16	80	0.2666
RAN (Platt 1991)	$\epsilon = 0.1$	57	342	0.378
	$\epsilon = 0.05$	92	552	0.376
	$\epsilon = 0.02$	113	678	0.373
	$\epsilon = 0.01$	123	738	0.374
RAN-GQRD (Rosipal *et al.* 1998)	$\epsilon = 0.1$	14	84	0.206
	$\epsilon = 0.05$	24	144	0.170
	$\epsilon = 0.02$	44	264	0.172
	$\epsilon = 0.01$	55	330	0.165
RAN-P-GQRD (Rosipal *et al.* 1998)	$\epsilon = 0.1$	14	84	0.206
	$\epsilon = 0.05$	24	144	0.174
	$\epsilon = 0.02$	31	186	0.160
	$\epsilon = 0.01$	38	228	0.183
Fuzzy Systems (Bersini *et al.* 1997)	Brute Force	10	190	0.1086
		11	206	0.1098
		12	228	0.1026
		13	247	0.2235
		14	266	0.1568
		15	285	0.1028
	Incremental	14	266	0.0965
Proposed Algorithm		13	78	0.2003 ± 0.0178
		14	84	0.1977 ± 0.0164
		15	90	0.1635 ± 0.0401
		16	96	0.1507 ± 0.0193
		17	102	0.1467 ± 0.0178
		18	108	0.1297 ± 0.0175
		19	114	0.1188 ± 0.0131
		20	120	0.1268 ± 0.0174
		21	126	0.1187 ± 0.0104
		22	132	0.1042 ± 0.0135
		23	138	0.1012 ± 0.0132
		24	144	0.0989 ± 0.0063
		25	150	0.0901 ± 0.0066

in Section VI where these expert mutation operators have been incorporated into a multiobjective evolutionary algorithm providing remarkable solutions for different approximation problems using really small population sizes.

Another important feature of the proposed approach is that it is able to find a set of Pareto-optimum solutions in only one execution. When the algorithms finishes, it returns a complete set of solutions with different compromises between the two objectives, while other approaches, which obtain only one solution per execution, have to be executed several times with different configurations to obtain separate solutions, thus, to make a fair comparison of the processing time, we should compare the processing time per solution found.

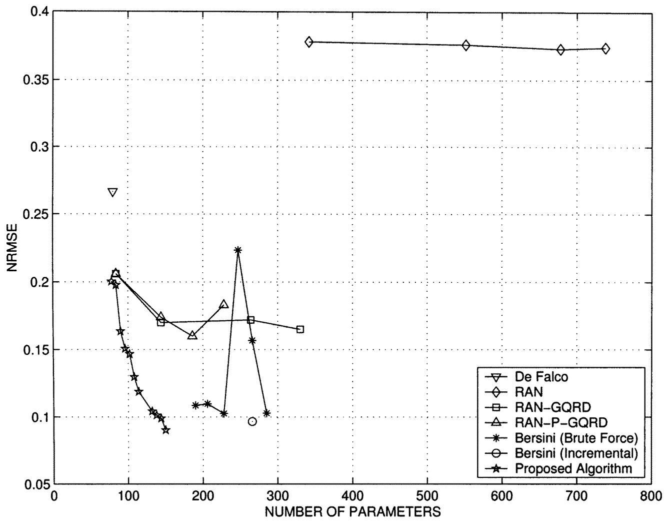

Fig. 18. Comparison of the proposed algorithm with others applied in the literature to predict the $s(t+85)$ value of the Mackey-Glass time series.

The algorithm proposed has also presented a robust behavior. The small standard deviations over the mean solutions show that it is able to find similar solutions starting from different random initial populations. It is also robust to noise in the training data, as shown in Section VI.

REFERENCES

[1] T. Bäck, Ed., *Proc. Seventh Int. Conf. Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann, 1997.

[2] *Handbook of Evolutionary Computation*, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds., Instit. Physics Publishing and Oxford University Press, Bristol, New York, MA, NY, 1997.

[3] H. Bersini, A. Duchateau, and N. Bradshaw, "Using incremental learning algorithms in the search for minimal and effective fuzzy models," in *Proc. 6th IEEE Int. Conf. Fuzzy Syst.*, Barcelona, Spain, July 1997, pp. 1417–1422.

[4] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[5] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford: Clarendon, 1995.

[6] R. Brent and F. Luk, "The solution of singular value and symmetric eigenvalue problems on multiprocessor arrays," *SIAM J. Sci. Stat. Comput.*, vol. 6, pp. 69–84, 1985.

[7] R. Brent, F. Luk, and C. Van Loan, "Computation of the singular value decomposition using mesh-connected processors," *J. VLSI Comput. Syst.*, vol. 1, pp. 242–270, 1984.

[8] D. S. Broomhead and D. Lowe, "Multivariate functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–355, 1988.

[9] L. Bruzzone and D. F. Prieto, "A technique for the selection of kernel-function parameters in RBF neural networks for classification of remote-sensing images," *IEEE Trans. Geosci. Remote Sensing*, vol. 37, no. 2, pp. 1179–1184, 1999.

[10] P. A. Castillo, "Optimización de perceptrones multicapa mediante algoritmos evolutivos," Ph.D. dissertation, Univ. Granada, Spain, Apr. 2000.

[11] P. A. Castillo, M. G. Arenas, J. G. Castellano, M. Cillero, J. J. Merelo, A. Prieto, V. Rivas, and G. Romero, "Function approximation with evolved multilayer perceptrons," in *2001 WSES Int. Conf. Neural Networks and Applications (NNA'01)*, N. E. Mastorakis, Ed.. Puerto De La Cruz, Tenerife, Canary Islands, 2001, pp. 195–200.

[12] V. Chankong and Y. Y. Haimes, *Multiobjective Decision Making Theory and Methodology*. New York: North-Holland, 1983.

[13] J. P. Charlier, M. Vanbegin, and P. V. Dooren, "On efficient implementations of kogbetliantz's algorithm for computing the singular value decomposition," *Numer. Math.*, vol. 52, pp. 279–300, 1988.

[14] S. Chen, S. A. Billings, and W. Luo, “Orthogonal least squares methods and their application to nonlinear system identification,” *Int. J. Contr.*, vol. 50, no. 5, pp. 1873–1896, 1989.

[15] S. Chen, C. F. N. Cowan, and P. M. Grant, “Orthogonal least squares learning algorithm for radial basis functions,” *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, 1991.

[16] S. Chen, P. M. Grant, and C. F. Cowan, “Orthogonal least squares learning algorithm for training multi-output radial basis function networks,” in *Proc. Inst. Elect. Eng.*, vol. 139, 1992, pp. 378–384.

[17] V. Cherkassky, D. Gehring, and F. Mulier, “Comparison of adaptive methods for function estimation from samples,” *IEEE Trans. Neural Networks*, vol. 7, pp. 969–984, July 1996.

[18] V. Cherkassky and H. Lari-Najafi, “Constrained topological mapping for nonparametric regression analysis,” *Neural Networks*, vol. 4, no. 1, pp. 27–40, 1991.

[19] A. G. Cunha, P. Oliviera, and J. Covas, Use of genetic algorithms in multicriteria optimization to solve industrial problems, pp. 682–688. In Bäck [1].

[20] I. De Falco, A. D. Cioppa, A. Iazzetta, P. Natale, and E. Tarantino, “Optimizing neural networks for time series prediction,” in *Proc. 3rd On-Line World Conf. Soft Computing (WSC3). Advances in Soft Computing—Eng. Design and Manufacturing*, R. Roy, T. Furuhashi, and P. K. Chawdhry, Eds, June 1998. Internet.

[21] K. Deb and D. E. Goldberg, “An investigation of niche and species formation in genetic function optimization,” in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed.. San Mateo, CA, 1989, pp. 42–50.

[22] R. E. Dennis and E. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[23] J. A. Dickerson and B. Kosko, “Fuzzy function approximation with ellipsoidal rules,” *IEEE Trans. Syst. Man Cyber. B*, vol. 26, pp. 542–560, Aug. 1996.

[24] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[25] C. M. Fonseca and P. J. Fleming, “Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization,” in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed., 1993, pp. 416–423.

[26] ——, “Multiobjective optimization and multiple constraint handling with evolutionary algorithms—Part I: A unified formulation,” *IEEE Trans. Syst. Man Cybern.*, vol. 28, no. 1, pp. 26–37, 1998.

[27] ——, “Multiobjective optimization and multiple constraint handling with evolutionary algorithms—Part II: Application example,” *IEEE Trans. Syst. Man Cybern.*, vol. 28, pp. 38–47, 1998.

[28] J. H. Friedman, “Projection pursuit regression,” *J. Amer. Statist. Assoc.*, vol. 76, pp. 817–823, 1981.

[29] ——, “Multivariate adaptive regression splines (with discussion),” *Ann. Statist.*, vol. 19, pp. 1–141, 1991.

[30] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[31] D. E. Goldberg and J. Richardson, “Genetic algorithms with sharing for multimodal function optimization,” in *Proc. 2nd Int. Conf. Genetic Algorithms*, J. J. Grefenstette, Ed.. Hillsdale, NJ, 1987, pp. 41–49.

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

[33] J. González, “Identificatión y optimización de redes de funciones base radiales para aproximación funcional,” Ph.D. dissertation, Univ. Granada, Spain, Sept. 2001.

[34] J. González, I. Rojas, H. Pomares, J. Ortega, and A. Prieto, “A new clustering technique for function approximation,” *IEEE Trans. Neural Networks*, vol. 13, pp. 132–142, Jan. 2002.

[35] J. González, I. Rojas, H. Pomares, M. Salmerón, and A. Prieto, “Evolution of fuzzy patches for function approximation,” in *Actas del X Congreso Español Sobre Tecnologías y Lógica Difusa, ESTYLF 2000*, A. Ollero, S. Sánchez, B. Arrue, and I. Baturone, Eds., Seville, Spain, Sept. 2000, pp. 489–495.

[36] A. E. Hans, “Multicriteria optimization for highly accurate systems,” *Multicriteria Optimization in Engineering and Sciences, Mathematical Concepts and Methods in Science and Engneering*, vol. 19, pp. 309–352, 1988.

[37] S. Haykin, *Neural Networks, a Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

[38] J. Hertz, A. Krough, and E. G. Palmer, *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison-Wesley, 1991.

[39] J. J. Holland, *Adaptation in Natural and Artificial Systems*: Univ. Michigan Press, 1975.

[40] C. L. Hwang and A. S. M. Masud, *Multiple Objective Decision Making—Methods and Applications, Volume 164 of Lecture Notes in Economics and Mathematical Systems*, Berlin: Springer-Verlag, 1979.

[41] *Proc. 5th IEEE Int. Conf. Fuzzy Syst.*, Sept. 1996. IEEE Neural Networks Council, Ed..

[42] H. Ishibuchi and T. Murata, “Multi-objective genetic local search algorithm,” *Proc. Third IEEE Int. Conf. Evolutionary Computation, ICEC’96*, pp. 119–124, 1996.

[43] J. S. R. Jang, “ANFIS: Adaptive network-based fuzzy inference system,” *IEEE Trans. Syst, Man, Cybern.*, vol. 23, pp. 665–685, May 1993.

[44] J. S. R. Jang and C. T. Sun, “Functional equivalence between radial basis function networks and fuzzy inference systems,” *IEEE Trans. Neural Networks*, vol. 4, pp. 156–159, Jan. 1993.

[45] P. P. Kanjilal and D. N. Banerjee, “On the application of orthogonal transformation for the design and analysis of feed-forward networks,” *IEEE Trans. Neural Networks*, vol. 6, pp. 1061–1070, 1995.

[46] N. B. Karayiannis, “Reformulated radial basis neural networks trained by gradient descent,” *IEEE Trans. Neural Networks*, vol. 10, pp. 657–671, May 1999.

[47] N. B. Karayiannis and G. W. Mi, “Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques,” *IEEE Trans. Neural Networks*, vol. 8, pp. 1492–1506, Nov. 1997.

[48] E. G. Kogbetliantz, “Diagonahzation of general complex matrices as a new method for solution of linear equations,” *Proc. Int. Congr. Math.*, vol. 2, pp. 356–357, 1954.

[49] ——, “Solution of linear equations by diagonalization of coefficients matrix,” *Quart. Appl. Math.*, vol. 13, pp. 123–132, 1955.

[50] L. Kuncheva, “Initializing of an RBF network by a genetic algorithm,” *Neurocomputing*, vol. 14, pp. 273–288, 1997.

[51] R. Langari, L. Wang, and J. Yen, “Radial basis function networks, regression weights, and the expectation-maximization algorithm,” *IEEE Trans. Syst. Man Cybern. A*, vol. 27, pp. 613–623, Sept. 1997.

[52] A. Lapedes and R. Farber, “How neural nets work,” *Neural Information Processing Systems*, pp. 442–456, 1988.

[53] M. C. Mackey and L. Glass, “Oscillation and chaos in physiological control systems,” *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[54] E. P. Maillard and D. Gueriot, “RBF neural network, basis functions and genetic algorithms,” *Proc. 1997 IEEE Int. Conf. Neural Networks*, vol. 4, pp. 2187–2190, 1997.

[55] D. W. Marquardt, “An algorithm for least-squares estimation of nonlinear inequalities,” *SIAM J. Appl. Math.*, vol. 11, pp. 431–441, 1963.

[56] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed: Springer-Verlag, 1996.

[57] J. E. Moody and C. J. Darken, “Fast learning in networks of locally-tuned processing units,” *Neural Computation*, vol. 1, pp. 281–294, 1989.

[58] G. C. Mouzouris and J. M. Mendel, “Designing fuzzy logic systems for uncertain environments using a singular-value-QR decomposition method,” *IEEE Neural Networks Council [41]*, pp. 295–301.

[59] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, “On the training of radial basis function classifiers,” *Neural Networks*, vol. 5, no. 4, pp. 595–603, 1992.

[60] V. Pareto, *Cours D’Economie Politique*, Lausanne: F. Rouge, 1896, vol. I and II.

[61] J. Park and I. W. Sandberg, “Universal approximation using radial-basis-function networks,” *Neural Computation*, vol. 3, pp. 546–257, 1991.

[62] ——, “Approximation and radial-basis-function networks,” *Neural Computation*, vol. 5, pp. 305–316, 1993.

[63] G. Patanè and M. Russo, “The enhanced-LBG algorithm,” *Neural Networks*, vol. 14, no. 9, pp. 1219–1237, 2001.

[64] J. Platt, “A resource allocation network for function interpolation,” *Neural Computation*, vol. 3, pp. 213–225, 1991.

[65] H. Pomares, “Nueva Metodología Para el Diseño Automático de Sistemas Difusos,” Ph.D. dissertation, Univ. Granada, Spain, Jan. 2000.

[66] H. Pomares, I. Rojas, J. González, and A. Prieto, “Structure identification in complete eule-based fuzzy systems,” *IEEE Trans. Fuzzy Syst.*, vol. 10, pp. 349–359, June 2002.

[67] H. Pomares, I. Rojas, J. Ortega, J. González, and A. Prieto, “A systematic approach to a self-generating fuzzy rule-table for function approximation,” *IEEE Trans. Syst., Man Cybern. B*, vol. 30, pp. 431–447, June 2000.

[68] I. Rojas, J. González, A. Cañas, A. F. Díaz, F. J. Rojas, and M. Rodriguez, “Short-term prediction of chaotic time series by using RBF network with regression weights,” *Int. J. Neural Syst.*, vol. 10, no. 5, pp. 353–364, 2000.

[69] I. Rojas, H. Pomares, J. González, J. L. Bernier, E. Ros, F. J. Pelayo, and A. Prieto, "Analysis of the functional block involved in the design of radial basis function networks," *Neural Processing Lett.*, vol. 12, no. 1, pp. 1–17, Aug. 2000.
[70] I. Rojas, H. Pomares, J. González, E. Ros, M. Salmerón, J. Ortega, and A. Prieto, "A new radial basis function networks structure: Application to time series prediction," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Networks*, S. I. Amari, C. L. Giles, M. Gori, and V. Piuri, Eds, Como, Italy: IEEE Computer Society, July 2000, vol. IV, pp. 449–454.
[71] I. Rojas, H. Pomares, J. Ortega, and A. Prieto, "Self-organized fuzzy system generation from training examples," *IEEE Trans. Fuzzy Syst.*, vol. 8, pp. 23–36, Feb. 2000.
[72] R. Rosipal, M. Koska, and I. Farkaš, "Prediction of chaotic time-series with a resource-allocating RBF network," *Neural Processing Lett.*, vol. 7, pp. 185–197, 1998.
[73] M. Salmerón, M. Anguita, M. Damas, C. Gil, and J. Ortega, "Predicción de series temporales en 'Clusters' de computadores," *Actas de las X Jornadas de Paralelismo*, pp. 13–15, Sept. 1999.
[74] M. Salmerón, J. González, J. Ortega, I. Rojas, and C. G. Puntonet, "Métodos ortogonales paralelos para predicción de series," in *Perspectivas en Paralelismo de Computadores. Actas de las XI Jornadas de Paralelismo*, J. Ortega, Ed, Granada, Spain: Univ. Granada, Sept. 2000, pp. 277–282.
[75] M. Salmerón, J. Ortega, and C. G. Puntonet, "On-line optimization of radial basis function networks with orthogonal techniques," in *Foundations and Tools for Neural Modeling, volume 1606 of Lecture Notes in Computer Science*, J. Mira and J. V. Sánchez-Andrés, Eds. New York: Springer-Verlag, 1999, pp. 465–478.
[76] M. Salmerón, J. Ortega, C. G. Puntonet, and A. Prieto, "Improved RAN sequential prediction using orthogonal techniques," *Neurocomputing*, vol. 41, no. 2, pp. 153–172, Aug. 2001.
[77] N. Srinivas and K. Dev, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolution. Computat.*, vol. 2, no. 3, pp. 221–248, 1995.
[78] T. Sudkamp and R. J. Hammell, "Interpolation, completion and learning fuzzy rules," *IEEE Trans. Syst. Man Cybern.*, vol. 24, pp. 332–243, Feb. 1994.
[79] M. Valenzuela-Rendón and E. Uresti-Charre, A non-generational genetic algorithm for multiobjective optimization, pp. 658–665. In Bäck [1].
[80] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst. Man Cybern.*, vol. 22, pp. 1414–1427, Nov. 1992.
[81] A. E. Webb and S. Shannon, "Shape-adaptive radial basis functions," *IEEE Trans. Neural Networks*, vol. 9, pp. 1155–1166, Nov. 1998.
[82] B. A. Whitehead, "Genetic evolution of radial basis function coverage using orthogonal niches," *IEEE Trans. Neural Networks*, vol. 7, pp. 1525–1528, 1996.
[83] B. A. Whitehead and T. D. Choate, "Evolving space-filling curves to distribute radial basis functions over an input space," *IEEE Trans. Neural Networks*, vol. 5, pp. 15–23, Jan. 1994.
[84] ——, "Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction," *IEEE Trans. Neural Networks*, vol. 7, no. 4, pp. 869–880, July 1996.
[85] J. Yen and L. Wang, "An SVD-based fuzzy model reduction strategy," *IEEE Neural Networks Council [41]*, pp. 835–841.
[86] ——, "Simplifying fuzzy rule-based models using orthogonal transformation methods," *IEEE Trans. Syst. Man Cybern. B*, vol. 29, pp. 13–24, Feb. 1999.

Jesús González was born in 1974. He received the M.Sc. degree in computer science in 1997 and the Ph.D. degree in 2001, both from the University of Granada, Granada, Spain.

He is currently an Assistant Professor with the Department of Computer Architecture and Computer Technology, University of Granada. His current areas of research interest are in the fields of function approximation using radial basis function neural networks, fuzzy systems, and evolutionary computation.

Dr. González received the Ph.D. Award for his dissertation from the University of Granada.

Ignacio Rojas received the M.Sc. degree in physics and electronics in 1992 and the Ph.D. degree in 1996, both from the University of Granada, Granada, Spain.

He was with the University of Dortmund, Germany, as an Invited Researcher from 1993 to 1995. In 1998 he was a Visiting Researcher with the BISC Group, University of California, Berkeley. He is currently an Associate Professor with the Department of Computer Architecture and Computer Technology, University of Granada. His research interests are in the fields of hybrid system and combination of fuzzy logic, genetic algorithms, and neural networks and financial forecasting.

Julio Ortega (M'98) received the B.Sc. degree in electronic physics in 1985, the M.Sc. degree in electronics in 1986, and the Ph.D. degree in 1990, all from the University of Granada, Granada, Spain.

He was with the Open University, U.K., and with the Department of Electronics, University of Dortmund, Germany, as an invited researcher. Currently, he is an Associate Professor with the Department of Computer Architecture and Computer Technology, University of Granada. His research interests include parallel processing and parallel computer architectures, artificial neural networks, and evolutionary computation. He has led research projects in the area of parallel algorithms and architectures for optimization problems.

Dr. Ortega's Ph.D. dissertation received the Ph.D. Award of the University of Granada.

Héctor Pomares was born in 1972. He received the M.Sc. degree in electronic engineering in 1995, the M.Sc. degree in physics in 1997, and the Ph.D. degree in 2000, all from the University of Granada, Granada, Spain.

He is currently an Associate Professor with the Department of Computer Architecture and Computer Technology, University of Granada. His current areas of research interest are in the fields of function approximation and on-line control using adaptive and self-organizing fuzzy systems.

Fco. Javier Fernández was born in Granada, Spain, in 1967. He received the M.Sc. and Ph.D. degrees in computer science from the University of Granada, in 1990 and 2001, respectively.

Since 1990, he has been a teaching assistant with the Department of Computer Architecture and Computer Technology, University of Granada. His research interests are bio-inspired neural systems and parallel computing.

Antonio Fco. Díaz received the M.Sc. degree in electronic physics and the Ph.D. degree in 2001, both from the University of Granada, Granada, Spain.

Currently, he is an Assistant Professor with the Department of Computer Architecture and Computer Technology, University of Granada. His interests include parallel processing and parallel computer and network architectures.