A Preliminary Study on Mutation Operators in Cooperative Competitive Algorithms for RBFN Design

María Dolores Pérez-Godoy, Antonio J. Rivera, Cristóbal J. Carmona and María José del Jesus

***Abstract*— Evolutionary Computation is a typical paradigm for the Radial Basis Function Network design. In this environment an individual represents a whole network. An alternative is to use cooperative-competitive methods where an individual is a part of the solution. CO^{2}RBFN is an evolutionary cooperative-competitive hybrid methodology for the design of Radial Basis Function Networks. In the proposed cooperative-competitive environment, each individual represents a Radial Basis Function, and the entire population is responsible for the final solution. In order to calculate the application probability of the evolutive operators over a certain Radial Basis Function, a Fuzzy Rule Based System has been used. In this paper, CO^{2}RBFN is adapted to the regression problem and an analysis of mutation operator is performed. To do so, two implementation of the mutation operator, based on gradient and based on clustering, have been implemented and tested. The results have been compared with other data mining and mathematical methods usually used in regression problems.**

I. Introduction

Evolutionary Computation (EC) [1] is a general stochastic optimization framework inspired by natural evolution. Typically, in this paradigm each individual represents a solution and is evaluated by means of a fitness value or a measure of the quality of the represented solution. New individuals are obtained using operators such as recombination or mutation and a selection mechanism, based on fitness, decide the individuals of the next generation.

Artificial Neural Networks (ANNs) [2] are another abstraction of a natural process. In short, an ANN is an interconnected network of very simple calculating units called neurons. Every connection in the network is assigned a weight which specifies the extent of possible influence. The structure of the network determines how the neurons influence each other. ANNs have successfully used in many applications of fields such as patter recognition, regression or time series prediction.

Unfortunately, determining the architecture and the parameters of a neural network is not an automatically process and classical methods can be trapped in local optimum [1]. Evolutionary algorithms (EAs) are typically used in ANN design [3]. EAs can quickly locate areas of high quality solutions when the domain is very large or complex. This is important in ANN design and training, where the search space is infinite, highly dimensional and multimodal.

Radial Basis Function Networks (RBFNs) are one of the most important ANN paradigms in the machine learning design field. An RBFN is a feed-forward ANN with a single layer of hidden units, called radial basis functions (RBFs). The first research on neural networks based on RBFs [4] was carried out at the end of the eighties. Since then, the overall efficiency of RBFNs has been proved in many areas like pattern classification [5], function approximation [6] and time series prediction [7]. The main features of a RBFN are a simple topological structure, with only one hidden layer, of neurons/RBFs that have a particular locally-tuned response that depends on the center and the width (radius) of each RBF. Also, they have universal approximation capability [6].

The objective of any RBFN design process is to determine centres, widths and the linear output weights connecting the RBFs to the output neuron layer. The most traditional learning procedure has two stages: first, unsupervised learning of centres and widths is used, and finally output weights are established by means of supervised learning. Clustering techniques [8] are normally used to adjust the centres. Regarding the widths, they may all be given the same value, may reflect the width of the previously calculated clusters (i.e., RBFs), or may be established as the average distance between RBFs, among other possibilities. In order to obtain the weights in the second stage, algorithms such as Least Mean Square (LMS) [9] or Singular Value Decomposition (SVD) [10] can be used.

One important paradigm for RBFN design is EC [16]. In most of the proposals within this evolutionary paradigm an individual represents a whole RBFN (Pittsburgh codification). An alternative to the classical Pittsburgh codification are the cooperative-competitive evolutionary or cooperative-coevolutive strategies [7] [11], where an individual of the population represents only a part of the solution.

The authors developed a hybrid cooperative-competitive evolutionary proposal for RBFN design, CO^{2}RBFN, applied to the classification problem [12]. In this paper, an adaptation of CO^{2}RBFN is presented in order to deal with the regression problem. In this adaptation process two biased mutation operators (based on gradient and based on clustering) have been implemented and tested, resulting in two version of CO^{2}RBFN: CO^{2}RBFN-Grad and CO^{2}RBFN-Clust respectively. The results obtained by these two versions have been compared with other data mining methods typically used for regression problems such as a fuzzy system developed with a GA-P algorithm (Fuzzy-GAP) [13], a multilayer perceptron network trained using a the well-known backpropagation learning algorithm (MLP-Back) [2], a classical design method for Radial Basis Function Network learning (RBFN-LMS) and a support vector machine based method (NU-SVR) [14]. Also a classical mathematical regression method, Linear-LMS [15]

Department of Computer Sciences, University of Jaen, Campus las Lagunillas s/n, 23071 Jaen, Spain; email: lperez, arivera, ccarmona, mjjesus(@ujaen.es).

978-1-4244-8126-2/10/$26.00

has been used in the comparison.

This paper is organized as follows: section 2 discusses generalities about RBFNs and reviews the RBFN evolutionary design. In section 3 CO^{2}RBFN is presented. The study and results obtained for regression problems are detailed in Section 4. In Section 5, conclusions and future works are outlined.

II. RBFN AND ITS EVOLUTIONARY DESIGN

From a structural point of view, an RBFN is a feed-forward neural network with three layers: an input layer with n nodes, a hidden layer with m neurons or RBFs, and an output layer with, in regression, one node (see figure 1).

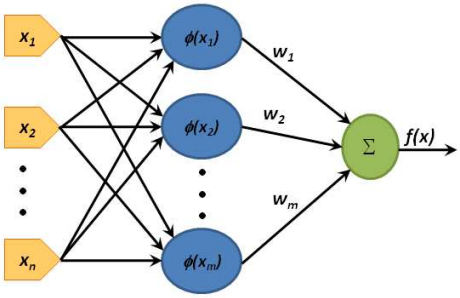

Fig. 1. RBFN Topology for time series forecasting

The m neurons of the hidden layer are activated by a radially-symmetric basis function, $\phi_i : R^n \rightarrow R$, which can be defined in several ways, being the Gaussian function the most widely used, i.e.: $\phi_i(\vec{x}) = \phi_i(e^{-(\|\vec{x}-\vec{c_i}\|/d_i)^2})$, where $\vec{c_i} \in R^n$ is the centre of basis function ϕ_i, $d_i \in R$ is the width (radius), and $\|\|$ is typically the Euclidean norm on R^n. This expression is the one used in this paper as the Radial Basis Function (RBF). The output nodes implement the following function:

$$f(\vec{x}) = \sum_{i=1}^{m} w_i \phi_i(\vec{x}) \quad (1)$$

As mentioned an important paradigm for the RBFN design is EC [16][5]. Typically, EC maintains a population of individuals (a whole RBFN), which evolves according to the operators as mutation, recombination or selection and each individual in the population receives a measure of its fitness in the environment. Nevertheless EC presents some difficulties for certain learning problems, especially in the evaluation of independent subcomponents (RBFs) [11].

In order to overcome these drawbacks a cooperative-competitive evolutionary strategy [7][11][17] can be used. This paradigm extends the basic computational model of evolution to provide a framework within which the individuals in the population represent only a part of the solution (a RBF in our case) and evolve in parallel, not only competing to survive but also cooperating in order to find a common solution (the whole RBFN) at the same time. This approach has the advantage of being computationally less complex, since an individual does not represent the whole solution but only a part of it. With this approach, two main problems must be addressed: credit assignment, or the fitness allocated to each individual according to its contribution to the final solution, and the mechanism used in order to maintain diversity among individuals of the population.

III. CO^{2}RBFN FOR REGRESSION

CO^{2}RBFN, is an evolutionary cooperative-competitive hybrid algorithm for the design of Radial Basis Function Networks (RBFNs).

In CO^{2}RBFN, each individual of the population represents, with a real representation, a basis function (RBF) and the entire population is responsible for the final solution. The individuals cooperate towards a definitive solution, but they must also compete for survival.

In this cooperative-competitive environment, in which the solution depends on the behaviour of many components, the fitness of each individual is known as credit assignment. In order to measure the credit assignment of an individual, three factors have been proposed to evaluate the role of each RBF in the network. These factors are: the RBF contribution to the network output, the error in the basis function radius, and the degree of overlapping among RBFs.

There are four evolutionary operators that can be applied to an RBF: an operator that eliminates the RBF, two operators that mutate the RBF, and finally an operator that maintains the RBF parameters These operators have been designed in order to adequately explore and exploit the search space.

The application of the operators is determined by a Fuzzy Rule-Based System (FRBS). The inputs of this system are the three parameters used for credit assignment and the outputs are the operators' application probability. To design the set of rules we must take into account the fact that an RBF is worse if its contribution is low, its error is high and its overlapping is also high, otherwise it is better. In this way the probability of eliminating an RBF is high when this RBF is worse and so on.

The main steps of CO^{2}RBFN, explained in the following subsections, are shown in the pseudocode in figure 2.

```
1. Initialize RBFN
2. Train RBFN
3. Evaluate RBFs
4. Apply operators to RBFs
5. Substitute the eliminated RBFs
6. Select the best RBFs
7. If the stop condition is not
     verified go to step 2
```

Fig. 2. Main steps of CO^{2}RBFN

A. RBFN initialization

To define the initial network, with a number of RBFs established by the size of the population, a simple process is used: a specified number, m, of neurons (i.e. the size of population) is randomly allocated among the different patterns

of the training set. To do so, each RBF centre, $\vec{c}_i$, is randomly established to a pattern of the training set. The RBF widths, d_i, will be set to half the average distance between the centres. Finally, the RBF weights, w_{ij}, are set to zero.

B. RBFN training

During this stage, RBF weights are trained. The Least Mean Square (LMS) algorithm [9] has been used to calculate the RBF weights. This technique exploits the local information that can be obtained from the behaviour of the RBFs. The equation shows the update of the weights.

$$\overline{w}_{k+1} = \overline{w}_k + \alpha \frac{e_k \overline{x}_k}{|\overline{x}_k^2|} \tag{2}$$

where k is the number of iteration, $\overline{w}_{k+1}$ is the next value of the weight vector, $\overline{w}_k$ is the present value of the weight vector, $\overline{x}_k$, in this case, is the RBFs output value of the actual input pattern vector. The present linear error, e_k, is defined as the difference between the desired output and the output network before adaptation. The α value is the *speed of learning*, it measures the size of the adjustment to be made. The choice of α controls stability and speed of convergence.

C. RBF evaluation

A credit assignment mechanism is required in order to evaluate the role of each basis function in the cooperative-competitive environment.

For an RBF ϕ_i, three parameters, a_i, e_i, o_i are defined:

- The contribution, a_i, of the RBF ϕ_i, $i = 1 \dots m$, is determined by considering the weight, w_i, and the number of patterns of the training set inside its width, npi_i. An RBF with a low weight and few patterns inside its width will have a low contribution:

$$a_i = \begin{cases} |w_i| & if \quad npi_i > q \\ |w_i| * (npi_i/q) & otherwise \end{cases} \tag{3}$$

  where q is the average of the npi_i values minus the standard deviation of the npi_i values.
- The error measure, e_i, for each RBF ϕ_i, is obtained by calculating the Mean Square Error (MSE):

$$e_i = \frac{\sum_{\forall pi_i} \big(f(pi_i) - y(pi_i)\big)^2}{npi_i} \tag{4}$$

  where $f(pi_i)$ is the output of the model for the point pi_i, inside the width of RBF ϕ_i, $y(pi_i)$ is the real output at the same point, and npi_i is the number of points inside the width of RBF ϕ_i.
- The overlapping of the RBF ϕ_i and the other RBFs is quantified by using the parameter o_i. This parameter is calculated by taking into account the fitness sharing [18] methodology, whose aim is to maintain diversity in the population. This factor is expressed as:

$$o_i = \sum_{j=1}^{m} o_{ij}$$

$$o_{ij} = \begin{cases} (1 - \|\phi_i - \phi_j\|/d_i) & if \|\phi_i - \phi_j\| < d_i \\ 0 & otherwise \end{cases} \tag{5}$$

  where o_{ij} measures the overlapping of the RBF ϕ_i y ϕ_j $j = 1 \dots m$.

D. Applying operators to RBFs

In this algorithm four operators have been defined in order to be applied to the RBFs:

- Operator Remove: eliminates an RBF.
- Operator Random Mutation: modifies the centre and width of an RBF. The width is altered with a probability inversely proportional to the number of features of the regression problem (n), at a percentage below 50% of the old width. The coordinates of the centre increase or decrease at a percentage below 50% of the width. The number of coordinates to be mutated is randomly obtained and is a number below 25% of the total number of coordinates.
- Operator Biased Mutation: modifies the width and all coordinates of the centre using local information of the RBF environment. As biased mutation operator, two proposals based on classical RBFN training methods, gradient and clustering, are described. The two proposals resulting in two implementations: CO^{2}RBFN-Grad and CO^{2}RBFN-Clust respectively and will be tested in the experimental section.

  a) *CO^{2}RBFN-Grad*. The operator used follows the recommendations of [19] that are similar to those used by the algorithm LMS algorithm. The error for the patterns within the radius of the RBF, ϕ_i, are calculated. For each coordinate of the center and the radius a value Δc_{ij} and Δd_i respectively are calculated. The new coordinates and the new radius are obtained by changing (increasing or decreasing) its old values to a random number (between 5% and 50% of its old width), depending on the sign of the value calculated.

$$\Delta d_i = \sum_k e(\overrightarrow{p_k}) \cdot w_i \tag{6}$$

  where $e(\overrightarrow{p_k})$ is the error for the pattern $\overrightarrow{p_k}$.

$$\Delta c_{ij} = sign(c_{ij} - p_{kj}) \cdot e(\overrightarrow{p_k}) \cdot w_i \tag{7}$$

  b) *CO^{2}RBFN-Clust*. As mentioned clustering techniques are habitually used in order to place RBFs inside the RBFN design [8]. The proposed mutation operator follows the basic method of the k-means clustering technique for calculating the center of the RBF. In this way the geometric center of the patterns inside the RBF width is calculated. The new coordinates of the RBF center are obtained by changing (increasing or decreasing) its old values in a random number (between 5% and 50% of its old width) in the direction of the mentioned geometric center, see equation 8.

$$\vec{C}_i = \frac{\sum_{i=1}^{n} \mu_{C_i}(\vec{pi}_i)\vec{pi}_i}{\sum_{i=1}^{n} \mu_{C_i}(\vec{pi}_i)} \quad (8)$$

where $\vec{pi}_i$ is a pattern within the radius of the RBF, ϕ_i and $\vec{C}_j$ is the calculated cluster for the RBF, ϕ_i. In order to establish the new radius, the distance to the farthest pattern inside is calculated. As above the new radius is obtained changing (increasing or decreasing) its old value in a random number (between 5% and 50% of its old width) in the direction of the mentioned farthest distance.

- Operator Null: in this case all the parameters of the RBF are maintained.

These mutation operators allow us to obtain an appropriate balance between exploitation and exploration, which is a desirable feature in every evolutionary algorithm. Biased mutations use local information from the RBF environment in order to achieve an optimal adaptation. On the other hand, random mutations carry out alterations that lead to the exploration of the environment and thus avoid local optimums.

The operators are applied to the whole population of RBFs. The probability for choosing an operator is determined by means of a Mandani-type fuzzy rule based system [20] which represents expert knowledge about the operator application in order to obtain a simple and accurate RBFN.

The inputs of this system are parameters a_i, e_i and o_i used for defining the credit assignment of the RBF ϕ_i. These inputs are considered as linguistic variables va_i, ve_i and vo_i. The outputs, p_{remove}, p_{rm}, p_{bm} and p_{null}, represent the probability of applying Remove, Random Mutation, Biased Mutation and Null operators, respectively. The number of linguistic labels has been empirically determined and the fuzzy sets have been defined according to their meaning. Figure 3 shows the membership functions for the input and output variables respectively. As defuzzification method the centre of area/gravity technique is used. Table I shows the rule base used to relate the described antecedents and consequents. In the table each row represents one rule. For example, the interpretation of the first rule is: If the contribution of an RBF is Low Then the probability of applying the operator Remove is Medium-High, the probability of applying the operator Random Mutation is Medium-High, the probability of applying the operator Biased Mutation is Low and the probability of applying the operator Null is Low.

The rule base represents expert knowledge, as mentioned, in the design of RBFNs. It was developed taking into account the fact that an RBF is worse if its contribution (a_i) is low, its error (e_i) is high and its overlapping (o_i) is also high. On the other hand, an RBF is better when its contribution is high, its error is low and its overlapping is also low. A worse RBF indicates that this neuron has problems performing a good role in its environment and therefore, important changes such as random mutations or even removing the RBF must be promoted. In these cases the probability of applying the biased mutation operator and the null operator is low. However, a better neuron implies that the RBF is working well in its environment. In these situations exploitation is promoted increasing the probability of applying the biased mutation operator. The probability of maintaining the neuron with the same parameters, applying the null operator, is also augmented. In these cases the probability of removing the RBF will be low. The probability of applying random mutation is usually high in order to promote a parsimonious evolution. It can be highlighted that this rule base represents general knowledge related with the design of RBFNs.

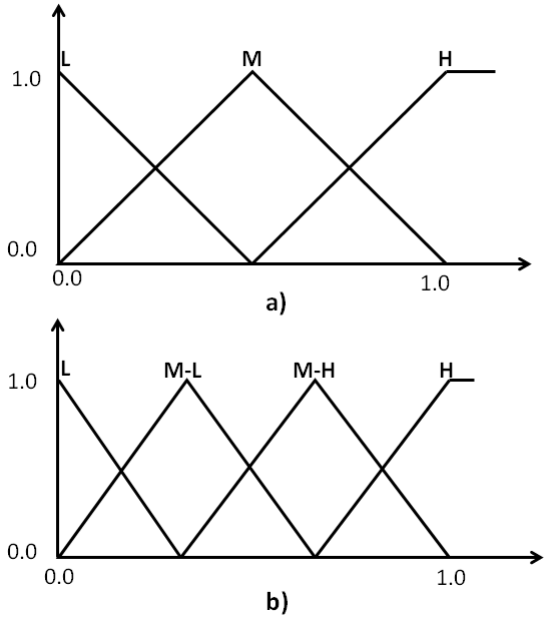

Fig. 3. a) input variables membership functions for the FRBS. b) output variables membership function

TABLE I

FUZZY RULE BASE REPRESENTING EXPERT KNOWLEDGE IN THE DESIGN OF RBFNS

	Antecedents			Consequents			
	v_a	v_e	v_o	p_{remove}	p_{rm}	p_{bm}	p_{null}
R1	L			M-H	M-H	L	L
R2	M			M-L	M-H	M-L	M-L
R3	H			L	M-H	M-H	M-H
R4		L		L	M-H	M-H	M-H
R5		M		M-L	M-H	M-L	M-L
R6		H		M-H	M-H	L	L
R7			L	L	M-H	M-H	M-H
R8			M	M-L	M-H	M-L	M-L
R9			H	M-H	M-H	L	L

E. Introduction of new RBFs

In this step of the algorithm, the eliminated RBFs are substituted by new RBFs. The new RBF is located in the pattern with maximum error of the training set or in a randomly chosen pattern with a probability of 0.5 respectively.

In the first instance, the new RBF is placed in the pattern with maximum error outside the radius of any RBF. The width of the new RBF will be set to the average of the RBFs in the population plus half of the minimum distance to the nearest RBF. Its weights are set to zero.

If it is chosen randomly, the RBF is located in the first pattern found outside any RBF width. The width of the new RBF is set to the average of the RBFs in the population and its weights are set to zero.

F. Replacement strategy

After applying the mutation operators, new RBFs appear. The algorithm uses the replacement scheme to decide which new RBFs will be included in the new population. To do so, for each mutated RBF a net, with the child RBF but without the parent RBF, is built. Then, every new net is evaluated (by means its training classification error) in order to determine the RBF (child or parent) with the best behaviour and to include it in the population.

IV. Experimental framework

In this study two versions of CO^{2}RBFN, four data mining methods and a classical mathematical method are applied to seven regression data-sets from the UCI repository [21]. Table II summarizes the data selected in this study and shows, for each data-set, number of attributes (#Attributes) and number of examples (#Examples).

TABLE II
DESCRIPTION DATA-SET

Name	#Attributes	#Examples
autompg8	7	392
daily-el.-en.	6	365
ele1	2	495
ele2	4	1056
forestfires	12	517
friedman	5	1200
machinecpu	6	209

As mentioned CO^{2}RBFN-Grad is the version of CO^{2}RBFN which a gradient-based operator as biased mutation operator and CO^{2}RBFN-Clust is the version of CO^{2}RBFN which a clustering-based operator as biased mutation operator.

In order to compare the results obtained by the different versions of CO^{2}RBFN, four data mining methods, habitually used in regression tasks have been chosen, such as a fuzzy system developed with a GA-P algorithm (Fuzzy-GAP) [13], a multilayer perceptron network trained using a the well-known backpropagation learning algorithm (MLP-Back) [2], a classical design method for Radial Basis Function Network learning (RBFN-LMS) and a support vector machine based method (NU-SVR) [14]. Also a classical mathematical regression method, Linear-LMS [15], has been used in the comparison:

- Fuzzy-GAP method [13]. A GA-P method [22] uses an evolutionary computation method, a hybrid between genetic algorithms and genetic programming, and optimized to perform symbolic regressions. Each element comprises a chain of parameters and a tree which describes a function, depending on these parameters. The two operators by means of which new members of the population are generated are crossover and mutation. In the GA-P algorithm both operations are performed independently over the tree and the parameter chain.
- MLP-BackProp. The multilayer perceptron (MLP) [2] network consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes. The input signal propagates through the network layer-by-layer. For the MLP, the most commonly activation functions, used in the computation nodes, are the logistic sigmoidal $\varphi(x) = 1/1 + e^{-x}$ or the hyperbolic tangent $\varphi(x) = \frac{1-e^{-x}}{1+e^{-1}}$. In regression, the net output is given by $f(\vec{x}) = \sum_i w_i \varphi_i$ where w_i are the weights to learn. In [23] is show that the MLP has universal approximation ability.
- RBFN-LMS. Builds an RBFN with a pre-specified number of RBFs. By means of the c-means clustering algorithm it chooses an equal number of points from the training set to be the centres of the neurons. Finally, it establishes a single radius for all the neurons as half the average distance between the set of centres. Once the centres and radio of the network have been fixed, the set of weights is analytically computed using the LMS algorithm [9].
- NU-SVR, the SVM (Support Vector Machine) model uses the sequential minimal optimization training algorithm and treats a given problem in terms of solving a quadratic optimization problem. The NU-SVR, called also v-SVM, for regression problems is an extension of the traditional SVM and it aims to build a loss function [14].
- Linear-LMS. In order to complete the experimentation an example of classical mathematical method for regression [15] based on gradient techniques have been chosen.

The implementation of these data mining methods has obtained from KEEL [24]. The main parameters used are set to the values indicated by the authors. It must be highlighted that in this version of KEEL, the complexity of the models obtained by the methods can not always be obtained. Therefore only can be managed the complexity of the MLP-BackProp method (30 neurons) and RBFN-LMS (50 RBFs/neurons). The parameters used for CO^{2}RBFN are shown in table III.

TABLE III
PARAMETERS USED FOR CO^{2}RBFN

Parameter	Value
Generations of the main loop	200
Number of RBF's	10

In order to estimate precision we use a ten-fold cross validation approach, 90% for training and 10% for testing. For each data-set we consider the average results of the Mean Square Error (MSE) of repeating five times the execution of

the ten partitions. Tables of results show the average and deviation of the obtained test errors.

Results of CO^{2}RBFN are shown in table IV. Results obtained by other data mining methods are shown in tables V and VI.

TABLE IV
MSE TEST ERROR OBTAINED BY CO^{2}RBFN-GRAD AND CO^{2}RBFN-CLUST

Data sets	CO^{2}RBFN-Grad	CO^{2}RBFN-Clust
autompg8	0.00602 ± 0.00220	0.00622 ± 0.00196
daily-el.-en.	0.00890 ± 0.00178	0.00895 ± 0.00187
ele1	0.00717 ± 0.00252	0.00749 ± 0.00369
ele2	0.00042 ± 0.00008	0.00049 ± 0.00015
forestFires	0.00356 ± 0.00574	0.00353 ± 0.00576
friedman	0.00601 ± 0.00117	0.00575 ± 0.00130
macninecpu	0.00515 ± 0.01008	0.00557 ± 0.01005

TABLE V
RESULTS OBTAINED BY FUZZY-GAP, LINEAR-LMS AND NU-SVR

Data sets	Fuzzy-Gap	Linear-LMS	NU-SVR
autompg8	0.01185 ± 0.00420	0.00895 ± 0.00296	0.00944 ± 0.00236
daily-el.-en.	0.01459 ± 0.00444	0.00922 ± 0.00197	0.00968 ± 0.00208
ele1	0.00870 ± 0.00304	0.00739 ± 0.00258	0.00911 ± 0.00257
ele2	0.00311 ± 0.00267	0.00037 ± 0.00003	0.00038 ± 0.00003
forestFires	0.00364 ± 0.00614	0.00342 ± 0.00578	0.00352 ± 0.00574
friedman	0.01639 ± 0.00631	0.00905 ± 0.00188	0.00932 ± 0.00155
macninecpu	0.00529 ± 0.00423	0.00480 ± 0.00358	0.00530 ± 0.00421

TABLE VI
RESULTS OBTAINED BY MLP-BACK AND RBFN-LMS

Data sets	MLP-Back	RBFN-LMS
autompg8	0.06328 ± 0.02044	0.00620 ± 0.00236
daily-el.-en.	0.04303 ± 0.01722	0.00952 ± 0.00265
ele1	0.34278 ± 0.06522	0.00696 ± 0.00235
ele2	0.18964 ± 0.07051	0.00022 ± 0.00007
forestFires	0.03136 ± 0.05045	0.00450 ± 0.00845
friedman	0.03638 ± 0.00701	0.00220 ± 0.00058
macninecpu	0.19421 ± 0.15188	0.00355 ± 0.00409

As analysis of the results, it can be deduced that the version of CO^{2}RBFN with a biased mutation operator based on gradient slightly outperforms, in accuracy, the version of CO^{2}RBFN with a biased mutation operator based on clustering. CO^{2}RBFN-Grad has a better (lower) error in five of the seven data-sets. The deviation of both implementations is similar and in the same line CO^{2}RBFN-Grad slightly outperforms CO^{2}RBFN-Clust. The fact that a gradient mutation operator works better than a clustering operator is according to [25] which concludes that, in regression problems, is better to take into account the output space (gradient operator) that the input space (clustering operator). In any case and from these results we can sentence that CO^{2}RBFN is a robust RBFN design methodology.

Regarding the accuracy obtained by the other methods, CO^{2}RBFN-Grad and CO^{2}RBFN-Clust clearly outperform methods such as Fuzzy-GAP, MLP-BackProp or NU-SVR, specialized data mining methods in the regression problem. Taking into account the complexity a model of CO^{2}RBFN has 10 neurons and a model of MLP-BackProp 30 neurons. CO^{2}RBFN-Grad slightly outperforms the accuracy of the Linear-LMS method achieving a best result in four of the seven data-sets. Linear-LMS obtain one more best result that CO^{2}RBFN-Clust, but when CO^{2}RBFN-Clust outperforms Linear-LMS its differences are higher. RBFN-LMS obtain models with 50 neurons versus the 10 neurons obtained for the CO^{2}RBFN models. Despiste this fact, CO^{2}RBFN-Grad obtains best results that RBFN-LMS in three data-sets (autompg8, daily-electric-energy and forestfires). In this sense must be highlighted the good job of CO^{2}RBFN placing and tuning the RBFs of the designed networks.

V. CONCLUSIONS

CO^{2}RBFN is a hybrid evolutionary cooperative-competitive algorithm for RBFN design and in this work it has been adapted to regression problems. An important key point of our proposal is the identification of the role (credit assignment) of each basis function in the whole network. In order to evaluate this value for a given RBF three factors are defined and used: the RBF contribution to the network's output, a_i; the error in the basis function radius, e_i; and the degree of overlapping among RBFs, o_i. In order to drive the cooperative-competitive process four operators are used: Remove, Random Mutation, Biased Mutation (based on clustering) and Null. The application of these operators is determined by a fuzzy rule-based system which represents expert knowledge of the RBFN design. The inputs of this system are the three parameters used for credit assignment.

In this adaptation process, we have developed two versions of CO^{2}RBFN: CO^{2}RBFN-Grad, a version where a gradient-based biased mutation operator has been implemented and CO^{2}RBFN-Clust a version where a clustering-based biased mutation operator has been implemented.

For comparisons, typical data mining methods applied for the regression problems have been executed. Concretely MLP-BackProp, a multilayer perceptron network trained with a backpropatation algorithm; Fuzzy-GAP, a fuzzy system developed with a GA-P algorithm; RBFN-LMS, a radial basis function network trained with the LMS algorithm; NU-SVR, a support vector machine based method and Linear-LMS a classical regression mathematical method.

From the results we can conclude CO^{2}RBFN-Grad slightly outperforms CO^{2}RBFN-Clust, may be because, in regression problems, it is better to take into account the output space that the input space.

Moreover CO^{2}RBFN outperforms methods as Fuzzy-GAP, MLP-BackProp or NU-SVR and CO^{2}RBFN-Grad slightly outperforms Linear-LMS. Even CO^{2}RBFN outperforms RBFN-LMS in three data-sets, taking into account that the models of RBFN-LMS have 50 neurons and the models of CO^{2}RBFN only have 10 neurons. In sort it can be said that CO^{2}RBFN is a robust RBFN design methodology and performs an adequate job placing and tuning the neurons or RBFs of the obtained

RBFN models. As future lines new designs of operators and application strategies will be carry out.

ACKNOWLEDGMENT

This work is supported by the Spanish Ministry of Science and Technology under the Projects TIN2008-06681-C06-02 and the Andalusian Research Plan TIC-3928.

REFERENCES

[1] T. Bäck, U. Hammel, and H. Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Transactions Evolutive Compututation*, 1(1):3–17, 1997.

[2] S. Haykin. *Neural Networks: A Comprehensive Foundation, 2nd Edition*. Prentice Hall, 1999.

[3] X. Yao. Evolving artificial neural networks. In *Proceedings of the IEEE*, volume 87, pages 1423–1447, 1999.

[4] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[5] O. Buchtala, M. Klimek, and B. Sick. Evolutionary optimization of radial basis function classifiers for data mining applications. *IEEE Transactions on System, Man, and Cybernetics, B*, 35(5):928–947, 2005.

[6] J. Park and I. Sandberg. Universal approximation using radial-basis function networks. *Neural Compututation*, 3(2):246–257, 1991.

[7] B. Whitehead and T. Choate. Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Transactions on Neural Networks*, 7(4):869–880, 1996.

[8] W. Pedrycz. Conditional fuzzy clustering in the design of radial basis function neural networks. *IEEE Transactions on Neural Networks*, 9(4):601–612, 1998.

[9] B. Widrow and M.A. Lehr. 30 years of adaptive neural networks: perceptron, madaline and backpropagation. In *Proceedings of the IEEE*, volume 78, pages 1415–1442, 1990.

[10] G. Golub and C. Van Loan. *Matrix computations*. Hopkins University Press. 3rd edition, 1996.

[11] M. Potter and K. De Jong. Cooperative coevolution: an architecture for evolving coadapted subcomponents. *Evolutionary Compututation*, 8(1):1–29, 2000.

[12] M. D. Pérez-Godoy, A. J. Rivera, F. J. Berlanga, and M. J. del Jesus. Co^{2}rbfn: An evolutionary cooperative-competitive rbfn design algorithm for classification problems. *Soft Computing*, 14(9):953–971, 2010.

[13] L. Sánchez and I. Couso. Fuzzy random variables-based modeling with ga-p algorithms. *In: B. Bouchon, R.R. Yager, L. Zadeh (Eds.) Information, Uncertainty and Fusion*, pages 245–256, 2000.

[14] R. E. Fan, P. H. Chen, and C. J. Lin. Working set selection using the second order information for training svm. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[15] J.S. Rustagim. *Optimization Techniques in Statistics*. Academic Press, 1994.

[16] C. Harpham, C.W. Dawson, and M.R. Brown. A review of genetic algorithms applied to training radial basis function networks. *Neural Computing and Applications*, 13:193–201, 2004.

[17] A.J. Rivera, I. Rojas, J. Ortega, and M.J. del Jesus. A new hybrid methodology for cooperative-coevolutionary optimization of radial basis function networks. *Soft Computing*, 11(7):655–668, 2007.

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

[19] J. Ghost, L. Deuser, and S. Beck. A neural network based hybrid system for detection, characterization and classification of short-duration oceanic signals. *IEEE Journal of Ocean Enginering*, 17(4):351–363, 1992.

[20] E. Mamdani. Applications of fuzzy algorithms for symple dinamycs plant. In *Proceedings of the IEEE*, volume 121, pages 1585–1588, 1974.

[21] A. Asuncion and D.J. Newman. Uci machine learning repository. *University of California, Irvine, School of Information and Computer Science*, 2007.

[22] L. Howard and D. D'Angelo. The ga-p: A genetic algorithm and genetic programming hybrid. *IEEE Intelligent Systems*, 10(3):11–15, 1995.

[23] K. Hornik, M. Stinchcombe, and H. Wite. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[24] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. Del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J.C. Fernández, and F. Herrera. Keel: A software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009.

[25] J. Gonzlez, I. Rojas, J. Ortega, and A. Prieto. A new clustering technique for function approximation. *IEEE Transactions on Neural Networks*, 13(1):132–142, 2002.