Soft Comput (2007) 11:655–668
DOI 10.1007/s00500-006-0128-9

ORIGINAL PAPER

A new hybrid methodology for cooperative-coevolutionary optimization of radial basis function networks

A. J. Rivera · I. Rojas · J. Ortega · M. J. del Jesus

Published online: 8 August 2006

Abstract This paper presents a new multiobjective cooperative–coevolutive hybrid algorithm for the design of a Radial Basis Function Network (RBFN). This approach codifies a population of Radial Basis Functions (RBFs) (hidden neurons), which evolve by means of cooperation and competition to obtain a compact and accurate RBFN. To evaluate the significance of a given RBF in the whole network, three factors have been proposed: the basis function's contribution to the network's output, the error produced in the basis function radius, and the overlapping among RBFs. To achieve an RBFN composed of RBFs with proper values for these quality factors our algorithm follows a multiobjective approach in the selection process. In the design process, a Fuzzy Rule Based System (FRBS) is used to determine the possibility of applying operators to a certain RBF. As the time required by our evolutionary algorithm to converge is relatively small, it is possible to get a further improvement of the solution found by using a local minimization algorithm (for example, the Levenberg–Marquardt method). In this paper the results of applying our methodology to function approximation and time series prediction problems are also presented and compared with other alternatives proposed in the bibliography.

A. J. Rivera (✉) · M. J. del Jesus
Department of Computer Science,
University of Jaén, Jaen, Spain
e-mail: arivera@ujaen.es
URL: http://wwwdi.ujaen.es/∼arivera

I. Rojas · J. Ortega
Department of Computer Architecture and
Computer Technology, University of Granada,
Granada, Spain

Keywords Cooperative–coevolution · Soft-computing techniques · RBF networks · Fuzzy rule based system · Function approximation

1 Introduction

The extensive research work carried out on the design of neural networks (Haykin 1999; Lipmann 1987; Platt 1991; Widrow and Lehr 1990), and more specifically on RBFNs, reveals the difficulty of this task and the absence of general mechanisms to automatically set the network parameters. In this field, the importance of soft-computing (Tettamanzi and Tomassini 2001) must be highlighted as one of the lines of development with the best results. Evolutionary Computation (EC) (Schwefel 1995) is one of the soft-computing strategies frequently applied to design neural networks.

According to Potter and De Jong (2000), it can be difficult to solve certain types of problems using evolutionary computation, especially when an individual represents a complete solution (i.e. a net) made of independent subcomponents. In these situations, the individuals' size can imply a premature convergence of the population and specific operators and mechanisms must be used to avoid it. Moreover, the role of good (or bad) independent subcomponents has not been taken into consideration in an individual solution or net. In theses cases, it is suitable to extend the basic computational model of evolution to provide a frame where the subcomponents evolve and cooperate to reach a solution in an independent way and to maintain the diversity.

A solution for this problem is Cooperative Coevolution (Potter and De Jong 2000; Rosin and Belew 1997), in

which the individuals in the population represent only a part of the solution and evolve in parallel, not only competing to survive but also cooperating to find a common solution at the same time. Compared with other evolutionary procedures, this new approach has the advantage of being less computationally complex and thus more cost effective since an individual does not represent the whole solution but only a part of it. The key point in a cooperative coevolutionary procedure is the *credit assignment* or the allocated fitness to each individual according to its contribution to the final solution.

Due to the novelty of this field and the difficulties of an adequate implementation of the above aspects, few proposals have been carried out in cooperative coevolution for the design of neural networks. A first example would be the Smalz and Conrad model (Smalz and Conrad 1994), where multilayer networks are built by subpopulations of nodes and the credit assignment is done according to the compatibility of each population with different networks. The Symbiotic Adaptive Neuro Evolution (SANE) developed by Moriarty and Miikkulainen (1997) is based on the coevolution of nodes, which take part in different candidate networks. The credit assignment to each node is done by using the average efficiency of the five best networks it participates in. Another interesting paper that applies cooperative coevolution to design neural networks can be shown in garcia et al. (2002). In this paper, subcomponent credit assignment is determined through a multiobjective cooperative coevolutionary technique which measures factors such as their efficiency and complexity.

In RBFNs, the response of the hidden neurons is localized, and they can be optimized by means of cooperative coevolutive methods. The main contribution of this paradigm is to direct the search using a credit assignment for each RBF based on its contribution to the performance of the network as a whole and also based on its importance.

In this paper a new hybrid cooperative coevolutive methodology for optimization of RBFNs is presented. To carry out the credit assignment three quality factors which define the role of each RBF have been defined: the basis function's contribution to the network's output; the error in the basis function radius; and the overlapping of RBFs. To combine them in a proper way a multiobjective technique is incorporated. Moreover, an FRBS is used to determine the application of different operators over an RBF.

The organization of this paper is as follows. Section 2 introduces the RBFNs and describes different alternatives for the optimization of the network parameters. In Sect. 3, the details of the proposed multiobjective cooperative coevolutionary algorithm are described.

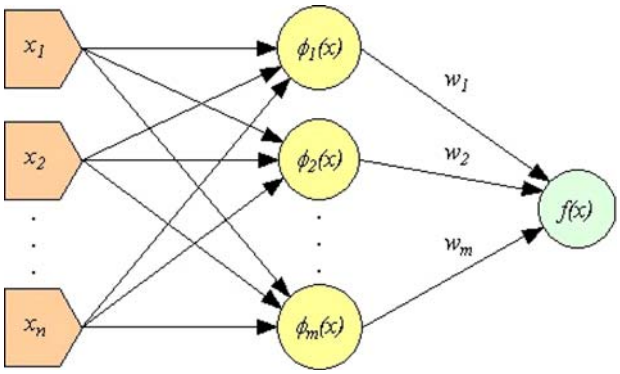

Fig. 1 Radial Basis Function Network

Section 4 describes the experiments and compares the results with other approaches presented in the bibliography. Finally, the conclusions of the paper are discussed in Sect. 5.

2 The design of radial basis function networks

Radial Basis Function Networks (RBFNs) (Broomhead and Lowe 1988; Powell 1985; Moody and Darken 1989) have been proved as a very suitable paradigm for function approximation problem. The function approximation learning problem consists in finding a function f that fits a set of sample data described by means of an input vector with n features and a target output.

RBFNs are a feedforward kind of neural network, with three layers: an input layer with n nodes, a hidden layer with m neurons or RBFs, and an output layer with one or several nodes. For function approximation problem, an RBFN has only one node in its output layer (see Fig. 1) and implements the function $f : R^n \rightarrow R$, that can be described by:

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

The m radially-symmetric radial basis functions, ϕ_i, are often taken to be translated dilations of a prototype radial basis function ϕ_i: $R^n \rightarrow R$ i.e. $\phi_i(\vec{x}) = \phi_i(\|\vec{x} - c_i\|/d_i)$, where $\vec{c}_i \in R^n$ is the center of basis function $\phi_i, d_i \in R$ is a dilation of scaling factor for the radius $\|\vec{x} - \vec{c}_i\|$, and $\|\|$ is typically the Euclidean norm on R^n. From all the possible choices for ϕ_i (Rojas et al. 1997), the Gaussian function $\phi(r) = \exp(-r^2)$, that is the most used, is the one used in this paper as RBF.

Different methods for the RBFNs learning have been set out in the specialized bibliography (Howlett and Jain 2001a, b provide a good overview of them). Evolutionary algorithms have been used for the design of RBFNs (Buchtala et al. 2005; Harpham et al. 2004; Lacerda et al. 2001, 2005 reviews can be found) but existing

approaches typically suffer from the problems of a high runtime and a premature convergence in local minima. Those are both objectives to reach in any evolutionary proposal for the design of RBFNs.

In Butchala et al. (2005) analyze the combination of evolutionary algorithms and RBFNs and consider that this hybridization can be classified in different categories related to the aspect that is optimized:

1. The computation of weights, i.e., centers and radii of RBFs and/or output weights.
2. The determination of the architecture of RBFN.
3. The feature selection for the RBFN.
4. The combination of networks in form of ensembles.

In this paper, the structure of the RBFN is fixed previoulsy and we only consider the evolutionary computation of centers and radii of RBFs. The weights are adjusted by means of Levenberg–Marquardt training. For this problem, in the specialized bibliography different evolutionary approaches have been presented with individuals which are complete RBFNs or single RBFs which constitute a network.

The former idea is investigated in Rivas et al. (2002), in which the genetic algorithm evolves a population of RBFNs to determine the network parameters. In this proposal the SVD algorithm (Golub and Van Loan 1996) was used to calculate the weights of the net. In a paper by González et al. (2003), this codification scheme is used in a multiobjective genetic algorithm which in some of its stages, also employs conventional techniques such as clustering algorithms, Orthogonal Least Squares OLS (Chen et al. 1999) or Singular Value Decomposition (SVD) (Golub and Van Loan 1996). In this approach the Levenberg-Marquardt algorithm (Marquardt 1963) is used to further minimize the approximation error.

The evolution of single RBFs is investigated by Whitehead and Choate (1996) where a cooperative-competitive evolution of centers and radii is proposed. In their genetic algorithm, selection operates on individual RBFs rather than on whole networks and the entire population corresponds to a single RBFN and the credit assignment to each individual is based on its contribution to the performance of the network as a whole and on the RBF weight. With this credit assignment strategy, it is shown that it is possible to maintain the diversity in the population. In this approach, the algorithm Singular Value Decomposition (SVD) (Golub and Van Loan 1996) is also applied to determine more accurate values for the weights of the network.

Although evolutionary alternatives to design RBFNs are frequent, few cooperative coevolutionary procedures have been implemented up to now, due to the problems arising in their development, especially with the credit assignment strategy which must promote competition among similar RBFs and cooperation among the different ones at the same time.

In this paper (an extended version can be found by Rivera 2003) we present a new hybrid methodology for the optimization of RBFNs based on a cooperative–coevolutionary algorithm. The proposal defines three criteria for the credit assignment, establishes a multiobjective selection for the individuals and uses an FRBS for the application of the operators in the evolutionary process. We have presented other coevolutionary approaches for this problem. In Rivera et al. (2003) the fitness of an RBF was calculated as a combination of different factors (a factor which takes into account its weight, a factor that measures its closeness to worst approximated points and a factor which quantifies the overlapping among RBFs), probabilistic rules are used to apply the operators and a monoobjective approach is considered. In the paper (Rivera et al. 2003), a preliminary approach was presented, with different criteria to evaluate an RBF.

3 A new approach for the optimization of RBF using a multiobjective cooperative–coevolutive algorithm

In the approach proposed in this paper each individual represents a basis function (RBF), and therefore forms only a part of the solution to the problem under analysis. Thus the entire population is responsible for the final solution. This allows for an environment where the individuals cooperate towards a definitive solution. However, they also compete for survival, since if their performance is poor, they will be eliminated.

One of the most important contributions of this proposal is the evaluation of each basis function by means of three criteria related with its role in the complete RBFN. These quality factors allow for an accurate evaluation of its participation within the network, or more generally, the assigned credit of each RBF to the final solution. To be precise, the three factors are the following ones:

- a_i: which measures the basis function's contribution to the network's output.
- e_i: that calculates the local error produced in the basis function radius.
- o_i: which evaluates the degree of overlapping of RBFs. This objective will be used to maintain the diversity of the population.

The three objectives are important to obtain accurate RBFNs, with a good local and global performance and

composed of a set of RBFs which describe information about different areas of search space.

Another important characteristic of our approach is that the way to combine these objectives is to evaluate the RBFs. In a previous approach (Rivera et al. 2001), we calculate the credit assignment as a weighted expression of characteristic parameters. This proposal has two disadvantages: the subjective determination of the weights and the possibility of obtaining a solution with a high value in one objective but very low values in the others. In this RBFN design process with a cooperative coevolutive approach it is most suitable to consider each objective in an independent way and to apply multiobjective theory to sort the RBFs.

To complete the hybrid bio-inspired architecture, a Fuzzy Rule Based System (FRBS) is used as the main strategy in order to decide the operators' application over a certain RBF. The inputs for the FRBS are the values of the objectives previously defined. The use of an FRBS for this task let us include in our cooperative coevolutive methodology the knowledge of an expert on the RBFN design.

The RBF network obtained from the hybrid bio-inspired procedure here described provides a solution to the problem of function approximation from a set of data points. This solution has a proper level of accuracy and simplicity (see the results in Sect. 4). Nevertheless, if more accuracy is required, it is possible to improve the quality of the solution found, by applying a local optimization procedure. More specifically, the Levenberg–Marquardt algorithm (Marquardt et al. 1963) has been incorporated to our cooperative coevolutionary procedure.

The fact that the cooperative coevolution methodology we propose, where a population of RBFs and not RBFNs evolve, decreases the computational cost must be highlighted (see experimental results in Sect. 4).

It is remarkable that in the evolutionary design of RBFNs (Buchtala et al. 2005) research field there are not so many coevolutionary approaches. The most important work is described before by Whitehead and Choate (1996), in which the credit assignment for each RBF is only based on the weight of the RBF. And, it does not include expert knowledge in the evolution.

In the following a detailed description of hybrid multiobjective methodology for cooperative coevolutionary optimization of RBFNs is shown.

3.1 Detailed algorithm

As is mentioned previously the complete population represents an RBFN with a number of radial basis functions determined by the population size.

Algorithm 1. Detailed algorithm for the design of the RBFNs

```
1. Initialize RBFN
2. Train RBFN
3. Evaluate RBFs
4. Select the worst RBFs
5. Apply operators to the worst RBFs using
   an FRBS
6. Substitute the RBFs that were eliminated
7. Train RBFN
8. If the stop − condition is not verified
   go to step 3, else go to 9
9. Apply Levenberg − Marquardt
```

The main steps of the proposed algorithm are shown in Algorithm 1. These steps are explained in the following subsections:

3.1.1 Initializing the population

A simple process is used to define the first network to begin with. According to the population size (i.e. the number of RBFs, m), the centre, $\bar{c}_i$, for each basis function, ϕ_i, in the network is randomly generated by keeping a minimum distance with the centres of the basis functions already generated. This minimum distance is the half of the radius that the m basis functions of the network would have if they were uniformly distributed and covering the whole input space without overlapping.

Initially, the radius will be set to a random value around the average distance from each RBF to the nearest neighbour RBF, and the weights are set to 0.

3.1.2 RBFN training

There are several methods like LMS (Widrow and Lehr 1990), SVD (Golub and Van Loan 1996), etc. to adapt the weights of a RBFN. In our proposal LMS is used because it is a simple and efficient (in computational cost) technique to calculate these weights. In addition, this technique can obtain a more or less precise solution, depending on the value of parameter alfa or the times that the method is applied. This is a suitable characteristic in our algorithm, because in the main loop we do not want a very accurate solution and we can save computational time. At the end of the algorithm we apply the Levenberg–Marquardt algorithm to reach a high quality solution.

In this stage, the LMS algorithm is executed over the training set until the error does not decrease for two consecutive iterations.

3.1.3 Evaluating RBFs

From the coevolutionary point of view of this work, a credit assignment mechanism is required to evaluate the role of each base function in the whole network. For this purpose, three parameters for each RBF ϕ_i are defined, taking into account the local behaviour of an RBF:

- a_i: measures the contribution of the base function to the output of the network.
- e_i: gives the error in the basis function radius.
- o_i: evaluates the overlapping among RBFs.

The contribution, a_i, of the RBF $\phi_i, i = 1, \ldots, m$, is determined by considering its weight w_i and the number of points of the training set inside its radius, pr_i.

$$a_i = \alpha \cdot |w_i| \cdot \frac{pr_i}{\frac{1}{m}\sum_{j=1}^{m} pr_j}, \tag{2}$$

where the parameter α allows the normalization of a_i inside the interval [0,1]. This normalization is required to assure an adequate FRBS performance.

The error measure, e_i, for each RBF ϕ_i, is obtained from the normalized root mean square error nrmse_i of the data points inside the radius of the RBF, as well as the standard deviation error, $S(\tilde{e}_i)$, also defined in the radius of the RBF.

$$e_i = \beta \cdot \text{nrmse}_i \cdot \frac{S(\tilde{e}_i)}{\frac{1}{m}\sum_{j=1}^{m} S(\tilde{e}_i)}. \tag{3}$$

Again, to be able to analyze the error values through an FRBS, they have to be moved around the interval [0, 1] with the parameter β.

Any overlapping among the RBF ϕ_i and other RBFs is quantified by using the parameter o_i. This parameter is calculated by taking into account the *fitness sharing* (Golberg and Richardson 1987) methodology, whose aim is to keep the diversity in the population, thus maintaining the bases of the coevolutionary algorithms.

The parameter o_i is expressed as:

$$o_i = \sum_{j}^{m} o_{ij}, \tag{4}$$

where o_{ij} measures the overlapping among the RBFs $\phi_j, j = 1, \ldots, m$, and ϕ_i, is calculated as:

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

where $||\cdot||$ is the Euclidean distance between the two basis functions, and d_i is the radius of the basis function ϕ_i.

In opposite to the calculation of the previous parameters, the expression to determine the values of o_i provides an absolute measure of the neurons overlapping, since it does not depend either on the problem, or on the current network. The parameters o_i can be directly analyzed using an FRBS without applying any kind of displacement.

Figure 2 shows the behaviour of the parameter o_{ij}. In a the value of o_{ij} is 0 because there is not overlap between i and j. However in b the value of o_{ij} is greater than 0, and therefore the overlap between the hidden neurons is detected.

3.1.4 Sorting and selecting basis functions

An appropriate way to sort the basis functions must take into account the three parameters that characterize their contribution to the final solution. A multiobjective (Coello et al. 2002) approach (not based in the search for the Pareto optimal front) has been used to take into account these three parameters, which have been considered as different objectives.

To sort the population, first the worst basis function ϕ_i is selected, which is the one with the worst values for the three parameters (the lowest value for a_i, the highest value for e_i, and the highest value for o_i). If there is no such basis function, the worst RBF is randomly chosen between the RBFs with the worst values in two of the three parameters. If such basis function does not exist, any of the parameters is randomly chosen and the basis function that presents the worst value within this parameter is selected as the worst basis function. This procedure is repeated with the remaining RBFs until all the basis functions have been sorted.

Once the sorting procedure is applied, the $m/2$ worst functions are selected and the operators described in the following subsection are applied to these selected RBFs.

3.1.5 Applying the operators to the set of worst basis functions

In the research field of cooperative coevolutive learning it is frequent to use only the mutation operator (Rechenberg 1973) to explore the search space. Moreover, in the evolutionary design of RBFNs the crossover operator can be destructive when the individuals are not made by independent subcomponents as in our case (Angeline et al. 1994; Yao 1999).

Taking this into account, in this proposal, only two operators are defined to be applied to the selected RBFs: an operator that eliminates RBFs, and an operator that adapts the selected RBF to the zone where it is situated. These operators exploit the local information that can

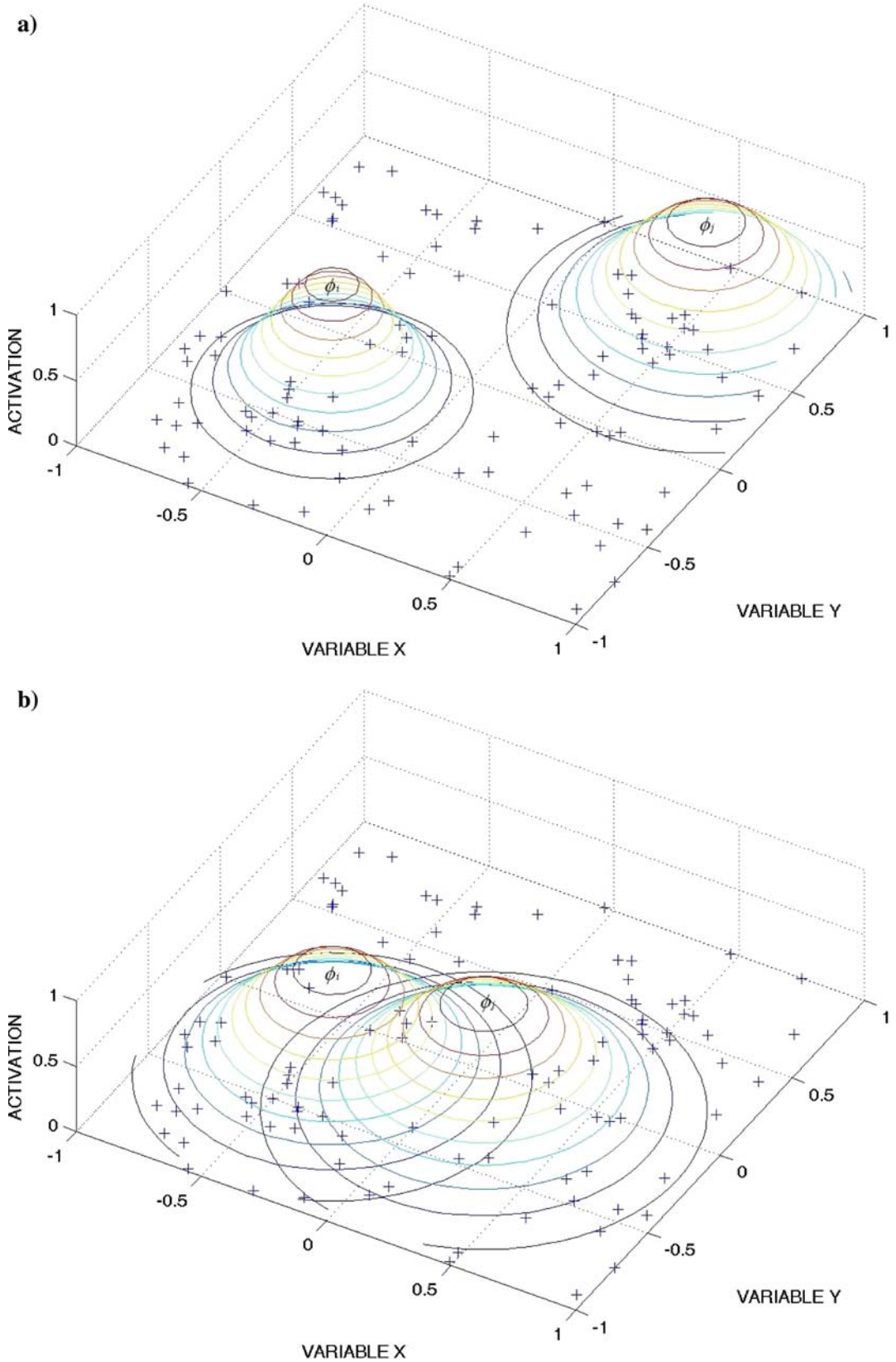

Fig. 2 Example of RBFs. **a** With no overlap. **b** With overlap

be obtained from the contribution of each basis function to the network. A more detailed explanation of the proposed operators is given below:

- *Operator REMOVE:* It simply eliminates an RBF.
- *Operator ADAPT:* This operator adapts the RBF to the examples belonging to the zone where the RBF is allocated. To do this, the output of the network is evaluated at the training points inside the radius of the RBF, and according to this output, the centre and the radius of the RBF are modified. These modifications are based on the LMS algorithm (Widrow and Lehr 1990).

As has been said, to decide the application of an operator to a basis function, an FRBS (Mendel 1995) is used. The goal of this fuzzy system is to take decisions about the operators according to expert knowledge and the behaviour of the RBF in the RBFN.

The three objectives previously defined characterize the contribution of each base function to the network, and its relation with the rest of the base functions. In this way, the parameters a_i, e_i and o_i are used as inputs in the fuzzy system. These inputs are considered as linguistic variables va_i, ve_i and vo_i. Moreover, two outputs are defined. The output p_{remove} is the probability of applying the operator REMOVE, while the output p_{adapt} determines the probability of applying the operator ADAPT.

The number of linguistic labels has been empirically determined and their fuzzy sets are defined according to their meaning. To be precise three linguistic labels Low, Medium, High are used for each input. Figure 3a, shows the membership functions for these linguistic labels. With respect to the outputs, four linguistic labels Low, Medium–Low, Medium–High, High are considered. Figure 3b shows the shape of the corresponding membership functions.

The rules base provides a set of simple guidelines from logics and heuristics that represent expert knowledge to be used in the design of RBFNs. To establish the set of rules the following facts are taken into account:

- 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, one RBF is better when its contribution is high, its error is low and its overlapping is also low.
- The probability to eliminate a basis function should increase, when the associated RBF is worse.
- The probability to adapt an RBF should increase, when the associated basis function is better.
- In general, the value of p_{adapt} should be high when the contribution is higher than an intermediate value.

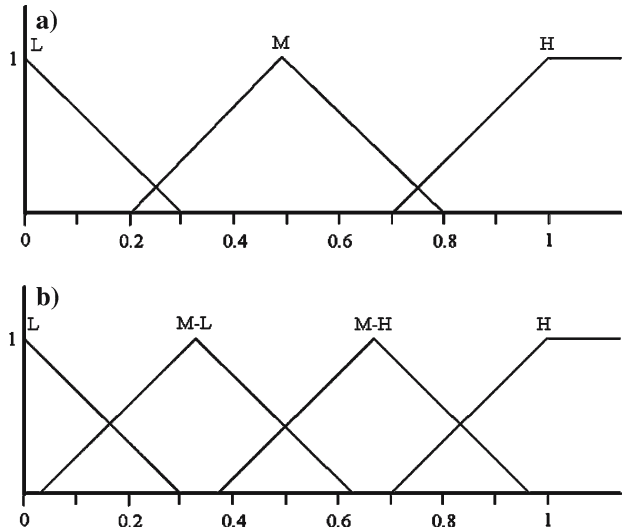

Fig. 3 **a** Input variables membership functions for the FRBS. **b** Output variables membership functions

Table 1 Rule base used in the FRBS

Antecedents			Consequents	
va	ve	vs	p_{remove}	p_{adapt}
L			M–H	M–H
M			M–L	H
H			L	H
	L		L	H
	M		M–L	H
	H		M–H	M–H
		L	L	H
		M	M–L	H
		H	M–H	M–H

Table 1 shows the simple rule base used, with a small number of rules which describe the expert knowledge.

These strategies are intended to keep a progressive network evolution. To do so, for most of the iterations, the adapt operator tries to adjust the RBF to the zone in which it is defined.

With regards to the inference engine, Mandani inference procedure is used (Mandani and Assilian 1975). This reasoning mechanism is configured considering the Max function as the T-conorm and the Min function as the T-norm and implication operators.

3.1.6 Introduction of new RBFs

In this step of the algorithm, the eliminated RBFs are substituted by new RBFs. These RBFs will be located in the zones with highest error, contrary to what is usually performed in this type of algorithm (Platt 1991), where the new RBFs are located at the points with highest error.

The above indicated zones with the highest error are determined by using the points that are outside any basis function radius. From these points, the one with the highest error will be the new centre of the one with a radius equal to the average of the RBFs radius. The error in this zone is calculated as the normalized root mean squared error (nrmse) of the points inside the zone.

The new RBFs will be iteratively introduced in these zones with higher error until the population is completed.

4 Experimental results

This section provides some experimental results obtained by the proposed algorithm in function approximation and time series forecasting problems.

For the experimentation the following conditions are established:

- The stop condition for the algorithm is a maximum number of iterations determined in a dynamical way: first, stop condition is set to 300 iterations. But if in the last 50 iterations an improvement of the error is obtained, the new stop condition is increased by 50.
- The solution is the RBFN chosen is the RBFN with the best error obtained during the entire run.
- The error used to measure the approximation of our model is the nrmse.
- The results are obtained after 30 runs of the method.

4.1 Function Approximation

The function approximation problem can be defined as the problem of estimating an unknown function, f, from a set of training examples: $(\vec{x}^k; z^k)$; $k = 1, 2, \ldots, K$; with $z^k = f(\vec{x}^k) \in R$, and $\vec{x}^k \in R^n$.

In the following subsections the functions used and the results obtained are shown.

Example 1 First, a 1-D target function is considered:

$$\text{dickerson}(x) = 3x(x-1)(x-1.9)(x+0.7)(x+1.8), \quad x \in [-2.1, 2.1]. \tag{6}$$

This function was proposed by Dickerson and Kosko (1996), where a hybrid neuro-fuzzy system is proposed using ellipsoidal rules to approximate the original function. This function was also used by Pomares et al. (2000), where a robust algorithm for the identification and optimization of a fuzzy system is proposed.

In this experimentation, to evaluate the behaviour of the proposed algorithm, a training set of 500 samples and a test set of 1,000 samples of the dickerson(x) function are generated by using inputs that have been randomly sampled from the interval where the function is defined as by González et al. (2003).

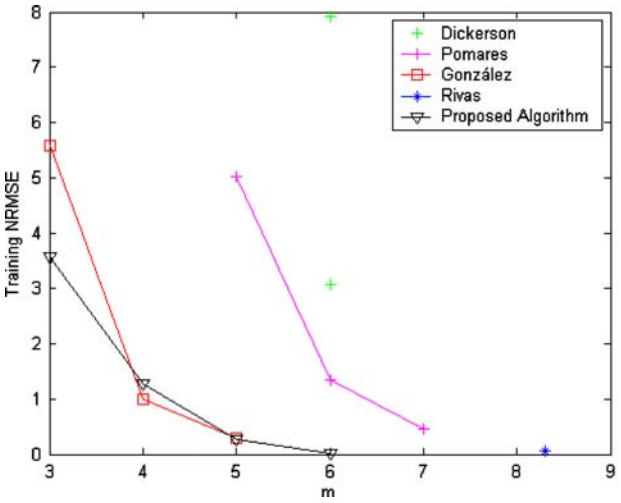

Fig. 4 Comparison of the results obtained for the Training NRMSE using different numbers of RBFs or Rules for the benchmark function *Dickerson*

Figure 4 and Table 2 show the results of the proposed algorithm in the approximation of dickerson*(x)* function and the comparison with other methods for direct synthesis of fuzzy systems and neural networks reported in the bibliography. The notation used is the following: m stands for the number RBFs or rules (depending on the model) in the solution obtained; n_p is the number of parameters learned by the algorithm; MSE stands for mean square error and NRMSE is the normalized root mean square error. In this case error for the training set is showed because only these results are found in the specialized bibliography. Anyway, we have checked that these results are very similar to test results for our algorithm.

From these data, it is clear that the inclusion of new RBFs decreases the error index. The proposed algorithm outperforms the results of other procedures presented by Dickerson and Kosko (1996), Pomares et al. (2000), Rives et al. (2002) and gives similar results to the error indexes by González et al. (2003). Nevertheless, the algorithm presented here is much simpler and has a lower computational cost (see results in Sect. 4.3) because (González et al. 2003) uses traditional evolutionary computation (an individual is a complete RBFN) or complex mutation operators based on Singular Value Decomposition (SVD) and Orthogonal Least Squares (OLS).

Example 2 In this example the bidimensional function approximation problem is considered. The two selected

Table 2 Comparison of the proposed algorithm with other methods for direct synthesis of neuro-fuzzy systems to approximate the target function dickerson (*x*)

Algorithm			*m*	n_p	MSE	NRMSE
Dickerson and Kosko (1996)	Different weights for rules	$w_k = 1$	6	–	94.65	–
		$w_k = 1/v_k$			28.25	–
		$w_k = 1/v_k^2$			10.53	–
	Not supervised				7.927	–
	Supervised				3.069	–
Pomares et al. (2000)			5	8	5.01	0.33
			6	10	1.35	0.17
			7	12	0.46	0.10
González et al. (2003)			3	9	5.57 ± 0.00	0.3455 ± 0.0000
			4	12	0.99 ± 0.49	0.1415 ± 0.0390
			5	15	0.30 ± 0.02	0.0797 ± 0.0023
Rivas et al. (2002)			8.3	24.9	0.05 ± 0.04	–
Proposed algorithm			3	9	3.57 + 0.00	0.2839 ± 0.0000
			4	12	1.27 ± 0.38	0.1671 ± 0.0285
			5	15	0.26 ± 0.09	0.0739 ± 0.0239
			6	18	5.7E-3 $\pm$ 0.01	0.0036 ± 0.0052

functions, f_4 and f_7, are defined by the following equations:

$$f_4(x_1, x_2) = \frac{1 + \sin(2x_1 + 3x_2)}{3.5 + \sin(x_1 - x_2)}, \quad x_1, x_2 \in [-2, 2], \tag{7}$$

$$f_7(x_1, x_2) = 1.9\left[1.35 + e^{x_1}\text{sen}(13(x_1 - 0.6)^2) \cdot e^{-x_2}\text{sen}(7x_2)\right] \quad x_1, x_2 \in [0, 1]. \tag{8}$$

f_4 function was reported by Cherkassky et al. (1996), where new methods for the design of Multilayer Perceptron are introduced. This function was also used by González et al. (2003), and Pomares et al. (2000).

The f_7 function is used by Charkassky and Lari-Najafi (1991) as a benchmark to compare the behaviour of several paradigms applied to function approximation, such as Projection Pursuit (PP) (Frideman 1981), Multivariate Adaptive Regression (MARS) (Friedman 1991), Constrained Topological Mapping (CTM) and Multilayer Perceptron (MLP). This function was also used as a benchmark by Castillo (2000), Cherkassky et al. (1996), and Pomares et al. (2000).

The training set is formed by 400 points randomly selected from each cell of a 20×20 grid partition of the input space. The test set is built with 961 points obtained by dividing the input interval into a 31×31 grid. The training and test sets are similar to the ones used by Cherkassky et al. (1996) and González et al. (2003).

Tables 3, 4 and Fig. 5 compare the proposed algorithm with other methods presented in the bibliography. It is important to note that the methodology presented by Pomares et al. (2000) gives an approximation with very low error due to the small number of non-linear parameters to be optimized. Indeed, there are a great number of linear parameters that can be optimally calculated in an FRBS. In this case, the results obtained by the proposed algorithm are similar to those presented by Pomares et al. (2000) and González et al. (2003) and better than those obtained in the rest of approaches.

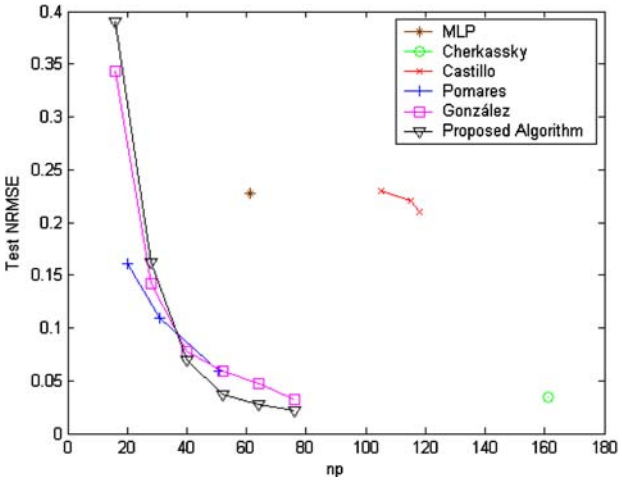

Fig. 5 Comparison of the result obtained for the Test NRMSE using different numbers of RBFs or Rules for the benchmark function *f*7

4.2 Time series prediction

Time-series prediction is an important practical problem which can be formulated as follows: given $x[t-(n-1)\Delta], x[t-(n-2)\Delta], \ldots, x[t-\Delta], x[t]$ determine $x[t+j]$, where n and j are fixed positive integers and Δ is

Table 3 Comparison of the proposed algorithm with other methods for direct synthesis of neuro-fuzzy systems to approximate the benchmark function f_4

Algorithm	m	n_p	Test NRMSE
Cherkassky et al. (1996)	40	161	0.052
Pomares et al. (2000)	42 (G)	64	0.061
	56 (G)	82	0.028
	81 (G)	113	0.015
González et al. (2003)	21	84	0.0473 ± 0.0086
	23	92	0.0362 ± 0.0088
	25	100	0.0265 ± 0.0033
	27	108	0.0239 ± 0.0054
	29	116	0.0214 ± 0.0049
Proposed algorithm	21	84	0.0485 ± 0.0065
	23	92	0.0437 ± 0.0056
	25	100	0.0351 ± 0.0078
	27	108	0.0299 ± 0.0057
	29	116	0.0252 ± 0.0046

Table 4 Comparison of the proposed algorithm with other methods for direct synthesis of neuro-fuzzy systems to approximate the benchmark function f_7

Algorithm	m	n_p	Test NRMSE
MLP (Cherkassky et al. 1996)	15	61	0.227
PP (Friedmann 1981)	–	–	0.206
CTM (Cherkassky et al. 1996)	–	–	0.197
MARS (Friedmann 1991)	–	–	0.179
Cherkassky et al. (1996)	40	161	0.034
Pomares et al. (2000)	16 (TP)	20	0.161
	25 (TP)	31	0.109
	42 (PT)	51	0.059
Castillo et al. (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
González et al. (2003)	7	28	0.1426 ± 0.0190
	10	40	0.0780 ± 0.0080
	13	52	0.0590 ± 0.0103
	16	64	0.0474 ± 0.0062
	19	76	0.0324 ± 0.0050
Proposed algorithm	7	28	0.1621 ± 0.0200
	10	40	0.0700 ± 0.0196
	13	52	0.0372 ± 0.0035
	16	64	0.0277 ± 0.0036
	19	76	0.0215 ± 0.0031

a lag time. One popular benchmark is the Mackey–Glass time series (Mackey and Glass 1977) which is generated from the following delay differential equation:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \frac{ax(t-\tau)}{1+x^{10}(t-\tau)} - bx(t). \tag{9}$$

Prediction of this time series is used as a benchmark for testing various neural-network architectures (De Falco et al. 1998; Platt 1991; Rosipal et al. 1998; Whitehead and Choate 1996). For the sake of comparison with earlier work, the following parameters have been selected: $\Delta = 6, j = 85, \tau = 17, a = 0.2$ and $b = 0.1$.

Four inputs: $x[t-18], x[t-12], x[t-6], x[t]$ have been considered, to forecast the output $x[t+85]$. Most of the authors who have used this time series predict in the long term $x[t+85]$, because this prediction problem is a significant challenge in which classical methods behave only slightly better than chance, thus, the use of RBFN is justified. A sample of the first 1,000 points was used in our study of the Mackey–Glass series. The first 500 points of the series are used as training data, and the final 500 points are used to validate the model.

As in the previous examples, Table 5 compares the results of our procedure with several approaches to solving this problem. When compared with algorithms that use the Multilayer Perceptron, as by De Falco et al. (1998), the results of our procedure surpass in general those obtained by De Falco et al. (1998).

Table 5 Comparison of the proposed algorithm with other methods for direct synthesis of neuro-fuzzy systems to approximate the benchmark Mackey–Glass time series prediction for $x(t + 85)$

Algorithm		m	n_p	Test NRMSE
MLP + BGA (De Falco et al. 1998)		16	80	0.2666
RAN (Platt 1991)	$\varepsilon = 0.1$	57	342	0.378
	$\varepsilon = 0.05$	92	552	0.376
	$\varepsilon = 0.02$	113	678	0.373
	$\varepsilon = 0.01$	123	738	0.374
RAN-GQRD (Rosipal et al. 1998)	$\varepsilon = 0.1$	14	84	0.206
	$\varepsilon = 0.05$	24	144	0.170
	$\varepsilon = 0.02$	44	264	0.172
	$\varepsilon = 0.01$	55	330	0.165
RAN-P-GQRD (Rosipal et al. 1998)	$\varepsilon = 0.1$	14	84	0.206
	$\varepsilon = 0.05$	24	144	0.174
	$\varepsilon = 0.02$	31	186	0.160
	$\varepsilon = 0.01$	38	228	0.183
Fuzzy System (Bersini et al. 1997)	Brute force	10	190	0.1086
		11	206	0.1098
		12	228	0.1026
		13	247	0.2235
		14	266	0.1586
		15	285	0.1028
	Incremental	14	266	0.0965
Whitehead and Choate (1996)		25	150	0.29
		50	300	0.18
		75	450	0.11
		125	750	0.05
González et al. (2003)		14	84	0.1977 ± 0.0164
		17	102	0.1467 ± 0.0178
		20	120	0.1268 ± 0.0174
		23	138	0.1012 ± 0.0132
		26	156	0.0999 ± 0.0192
		29	174	0.0891 ± 0.0131
Rivas et al. (2002)		72	432	0.177 ± 0.004
Proposed algorithm		14	84	0.1675 ± 0.0210
		17	102	0.1281 ± 0.0091
		20	120	0.1133 ± 0.0125
		23	138	0.1059 ± 0.0134
		26	156	0.0947 ± 0.0101
		29	174	0.0803 ± 0.0066

Other comparisons can be done with methodologies based on RBFNs, such as the classical model RAN (Platt 1991), which iteratively builds an RBFN by analyzing the novelty of the input data. Although the algorithm RAN was improved by Rosipal et al. (1998), giving as the result, the algorithms RAN-GQRD and RAN-P-GQRD, the proposed algorithm improves these methods.

Another interesting algorithm to consider is the one presented by Whitehead and Choate (1996). This algorithm evolves a population of individuals where each one of these individuals represents an RBF. Therefore, this algorithm corresponds to an approach similar to our procedure, keeping in mind that it also carries out a final minimization of the error by the application of the SVD algorithm. Comparing the NRMSE, one can observe that, although the algorithm offers similar error values to those obtained by our algorithm, it requires a higher number of RBFs. Regarding the computational cost, both algorithms can be considered similar.

With regard to the algorithm (González et al. 2003), the differences in error are small; nevertheless it is important to take into account that the computational cost of this algorithm is very high compared with the cost of our algorithm.

The results of our procedure are also compared with those of an algorithm based on a fuzzy system presented by Bersini et al. (1997). In this paper two different algorithms are presented in order to optimize the membership functions and the fuzzy system generated: a brute force one, and an incremental one. The unstable behaviour of the brute force is also evident when the number

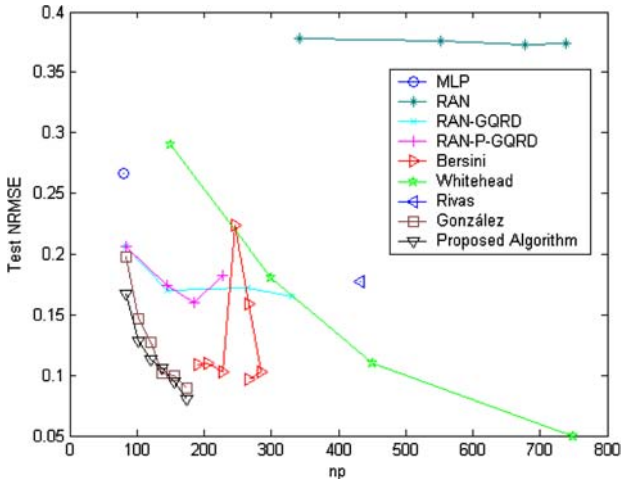

Fig. 6 Results obtained by different algorithms in the prediction of the long term, $x(t+85)$, Mackey–Glass time series

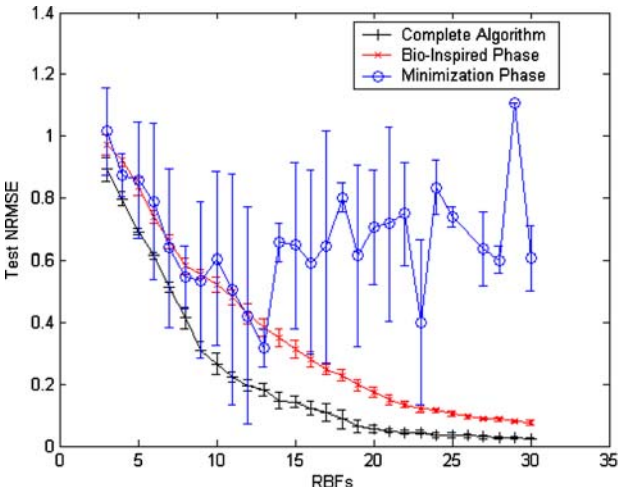

Fig. 7 Comparison of the results of the different phases of the algorithm for the approximation of the function f_4. + results of the complete algorithm. x results if only Bio-inspired phase is applied. o results if only minimization phase is applied

of fuzzy rules increases. It is important to note that, although there is a certain similarity among the errors obtained with those of our procedure, the number of free parameters of the fuzzy models generated by this method Bersini et al. (1997) is far greater.

Finally, it is important to note that our procedure is able to find a set of Pareto-optimum solutions that dominate all the solutions in the Table 5. This is graphically shown in Fig. 6.

4.3 Analysis of the independent phases and computational cost of the algorithm

Next, the two main phases of the algorithm, the hybrid bio-inspired phase and the minimization phase, are analyzed. For this objective both the bio-inspired phase and the minimization phase are executed from the same initialization and the results of the two phases of the algorithm and the results of the complete algorithm are shown in the Fig. 7.

As can be seen from these data, the bio-inspired phase reaches a good solution near the solution of the complete algorithm. Also, this phase has coherent behaviour and when new RBFs are introduced the RBFN error decrease. Moreover the standard deviation is low. If only the minimization phase is applied the results are very bad with incoherent behaviour, where more RBFs do not imply a better result or where the standard deviation in high.

These results also confirm the reliability of the complete algorithm. The solutions obtained from the hybrid bio-inspired phase are the most suitable because with a high probability, the minimization phase reaches practically a global optimum.

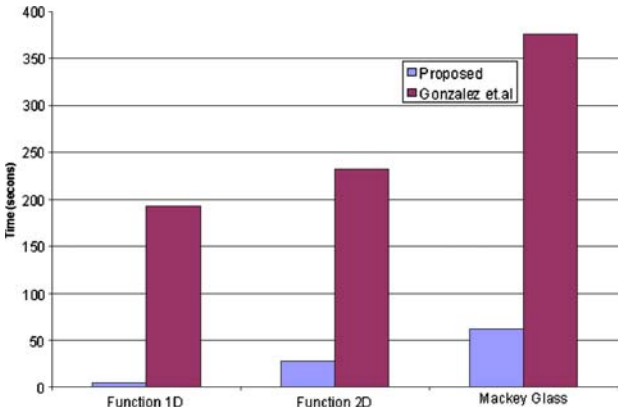

Fig. 8 Comparison of the mean computational time required for the proposed algorithm for approximating different problems (function of one and two dimensions, the Mackey–Glass time series), compared with the approach presented by Gonzalez et al. (2003)

Finally, one of the most important feature of the proposed methodology is the computational cost of the algorithm. In fact, the presented algorithm is very fast because the cooperative coevolutionary method selection operates on individual RBFs rather than on whole networks, therefore instead of evolving complex neural networks, it evolves individual neurons. The entire population corresponds to a single RBFN, instead of different approaches (González et al. 2003), in which the entire population corresponds to a large number of neural networks. This can be verified analyzing the computational time in Fig. 8.

5 Conclusions

A new multiobjective cooperative–coevolutionary algorithm for the optimization of the parameters defining an RBFN has been proposed. An important contribution of the presented procedure is the identification of the role of each basis function defining the network. In order to evaluate the significance of a given RBF, three factors are 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. Two main operators are used to drive the cooperative–coevolutive algorithm: one is used for the elimination of a hidden neuron and the other for the adaptation of the parameters defining the neurons. In this way, our hybrid procedure includes a fuzzy rule based system to decide the application of the operators to a given RBF (to remove or adapt it). Thus, the probability of applying an operator is provided by the fuzzy rule based system by using as an input the three parameters, a_i, e_i, and o_i, used for credit assignment.

The hybrid procedure here proposed is able to find a RBF network composed of few RBFs and with high accuracy. Nevertheless, it is also possible to include a further step if a further improvement in the quality of the RBF should be obtained. In our present implementation this step uses the Levenberg–Marquardt method as a local minimization algorithm that makes it possible to obtain the local minimum near the solution given by our bio-inspired procedure. In this way, the initial conditions given to Levenberg–Marquardt method are the most suitable ones due to the quality of the cooperative coevolutionary algorithm used in the first phase, thus implying that the final local optimum obtained will be a global optimum with a high degree of probability.

The paper provides a detailed comparison of our algorithm and other solutions presented in the bibliography for function approximation and time series prediction. From the analysis of the results obtained, it can be concluded that the proposed algorithm produces in general, good results for function approximation, is much simpler and has a lower computational cost than other multiobjective evolutionary algorithms presented in the bibliography that use traditional evolutionary computation and complex operators. It is important to take into account the low values for the standard deviations of the error index. This circumstance indicates the robustness of the presented procedure with respect to its error indices. Moreover, as the differences between the training set error and the test set errors are small, the good generalization capability of our algorithm is clearly demonstrated.

Finally as regards to future work, it would be interesting to analyze the importance and the contribution of each objective in the final solution and as a future line, a deeper research in the evolution of the individuals applying multiobjective techniques will be carried out.

Acknowledgments This work has been partially supported by the CICYT Spanish Projects TIN2004-01419, TIC2002-04036-C05-04 and TIN2005-04386-C05-03.

References

Angeline P, Saunders G, Pollack J (1994) An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans Neural Netw 5(1):54–65

Bersini H, Duchateau A, Bradshaw N (1997) Using incremental learning algorithms in the search for minimal and effective models. In: Proceedings of the 6th IEEE international conference on fuzzy system. IEEE Computer Society Press, Barcelona, pp 1417–1422

Broomhead D, Lowe D (1988) Multivariable functional interpolation and adaptive networks. Complex Syst 2:321–355

Buchtala O, Klimek M, Sick B (2005) Evolutionary optimization of radial basis function classifiers for data mining applications. IEEE Trans Syst Man Cybern B 35(5):928–947

Castillo P (2000) Optimización de perceptrones multicapa mediante algoritmos evolutivos. Ph.D. dissertation, University of Granada, Spain

Chen S, Woo Y, Luk B (1999) Combined genetic algorithms optimization and regularized orthogonal least squares learning for radial basis function networks. IEEE Trans Neural Netw 10(5):1239–1243

Cherkassky V, Lari-Najafi H (1991) Constrained topological mapping for non-parametric regression analysis. Neural Netw 4(1):27–40

Cherkassky V, Gehring D, Mulier F (1996) Comparison of adaptive methods for function estimation from samples. IEEE Trans Neural Netw 7(4):969–984

Coello CA, Van Veldhuizen DA, Lamont GB (2002) Evolutionary algorithms for solving multi-objective problems. Kluwer, New York

De Falco I, Della Cioppa A, Iazzetta A, Natale P, Tarantino E (1998) Optimizing neural networks for time series prediction. In: Roy R et al (eds) Proceedings of the 3rd on line world conference on soft computing (WSC3). Advances in Soft Computing – Engeenering design and Manufacturing Internet

Dickerson J, Kosko B (1996) Fuzzy function approximation with ellipsoidal rules. IEEE Trans Syst Man Cybern B 26(4):542–560

Friedman J (1981) Projection pursuit regression. J Am Stat Assoc 76:817–823

Friedman J (1991) Multivariate adaptive regression splines (with discussion). Ann Stat 19:1–141

García N, Hervás C, Muñoz J (2002) Multi-objective cooperative coevolution of artificial neural networks. Neural Netw 15:1259–1278

Goldberg D, Richardson J (1987) Genetic algorithms with sharing for multimodal function optimization. In: Grefenstette J (ed) Proceedings of the second international conference on genetic algorithms. Lawrence Erlbaum Associates, pp 41–49

Golub G, Van Loan C (1996) Matrix computations, 3rd edn. Johns Hopkins University Press, Baltimore

González J, Rojas I, Ortega J, Pomares H, Fernández F, Diaz A (2003) Multiobjective evolutionary optimization of the size, shape and position parameters of radial basis function networks for function approximation. IEEE Trans Neural Netw 14(6):1478–1495

Harpham C,Dawson C, Brown M (2004) A review of genetic algorithms applied to training radial basis function networks. Neural Comput Appl 13:193–201

Haykin S (1999) Neural Networks. A comprehensive foundation, 2nd edn. Prentice-Hall, Upper Saddle River

Howlett RJ, Jain LD (2001a) Radial basis function networks 1 — recent developments in theory and applications. Studies in fuzzyness and soft computing. Physica-Verlag, New York

Howlett RJ, Jain LD (2001b) Radial basis function networks 2 — New avances in design. Studies in fuzzyness and soft computing. Physica-Verlag, New York

Lacerda E, de Carvalho A, Ludermir T (2001) Evolutionary optimization of RBF networks. In: Howlett RJ, Jain LC (eds) Radial basis. function networks 1 — Recent developments in theory and applications, vol 66.Studies in fuzzyness and soft computing. Physica-Verlag, New York, pp 281–309

Lacerda E, Carvalho A, Padua, A, Bernarda T (2005) Evolutionary radial basis functions for credit assessment. Appl Intell 22:167–181

Lipmann R (1987) An introduction to computing with neural nets. IEEE Trans Acoust Speech Signal Process 2(2):4–22

Mackey M, Glass L (1977) Oscillation and chaos in physiological control system. Science 197:287–289

Mandani E, Assilian S (1975) An experiment in linguistic synthesis with a fuzzy logic controller. Int J Man-Mach Stud 7(1):1–13

Marquardt DW (1963) An algorithm for least-squares estimation of non-linear inequalities. SIAM J Appl Math 11:431–441

Mendel J (1995) Fuzzy logic system for engineering: a tutorial. Proc IEEE 83(3), pp 345–377

Moody J, Darken CJ (1989) Fast learning in networks of locally-tuned processing units. Neural Comput 1:281–294

Moriarty D, Miikkulainen R (1997) Forming neural networks through efficient and adaptive coevolution. Evol Comput 5(4):373–399

Platt J (1991) A resource-allocating network for function interpolation. Neural Comput 3:213–225

Pomares H, Rojas I, Ortega J, González J, Prieto A (2000) A systematic approach to self-generating fuzzy rule-table for function approximation. IEEE Trans Syst Man Cybern B 30(3):431–447

Potter M, De Jong K (2000) Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evol Comput 8(1):1–29

Powell M (1985) Radial basis functions for multivariable interpolation: a review. In: IMA. Conference on Algorithms for the approximation of functions and data, pp 143–167

Rechenberg I (1973) Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart

Rivas V, Castillo PA, Merelo-Guervós JJ (2002) Evolved rbf networks for time-series forecasting and function approximation. LNCS 2439:505

Rivera AJ, Ortega J, Prieto A (2001) Design of rbf networks by cooperative/competitive evolution of units. In: International conference on artificial neural networks and genetic algorithms. ICANNGA 2001, pp 375–378

Rivera AJ, Ortega J, Rojas I, del Jesus M (2003) Co-evolutionary algorithm for RBF by self-organizing population of neurons. LNCS 2686:470–477

Rivera AJ (2003) Diseño y optimización de redes de funciones de base radial mediante técnicas bioinspiradas. Ph.D. dissertation, University Granada, Spain

Rojas I, Valenzuela O, Prieto A (1997) Statistical analysis of the main parameters in the definition of radial basic function networks. LNCS 1240:882–891

Rosin C, Belew R (1997) New methods for competitive coevolution. Evol comput 5(1):1–29

Rosipal R, Koska M, Farkaš I (1998) Prediction of chaotic time series with a resource allocating RBF networks. Neural Process Lett 7:185–197

Schwefel HP (1995) Evolution and optimum seeking. Wiley, New York

Smalz R, Conrad M (1994) Combining evolution with credit apportionment: a new learning algorithm for neural nets. Neural Netw 7(2):341–351

Tettamanzi A, Tomassini M (2001) Soft computing. Integrating evolutionary, neural and fuzzy system. Springer, Berlin Heidelberg New York

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

Widrow B, Lehr MA (1990) 30 Years of adaptive neural networks: perceptron, madaline and backpropagation. Proc IEEE 78(9):1415–1442

Yao X (1999) Evolving artificial neural networks. Proc IEEE 87(9):1423–1447