Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

# Multiobjective Cuckoo Search Applied to Radial Basis Function Neural Networks Training for System Identification

**Helon Vicente Hultmann Ayala** *
**Leandro dos Santos Coelho**[*, **]

[*] *Industrial and Systems Engineering Graduate Program (PPGEPS)*
*PUCPR, Curitiba, Brazil.*
[**] *Electrical Engineering Graduate Program (PPGEE)*
*UFPR, Curitiba, Brazil.*
*e-mails: {helon.ayala; leandro.coelho}@pucpr.br*

**Abstract:** In the present work we introduce a system identification framework where no a priori information on the system to be identified is available. Focusing in the specific case of radial basis functions neural networks models, we insert the choice of the model complexity and its inputs in the optimization procedure together with the model parameters, aiming at accuracy, model validity and regularization in a multiobjective approach. The multicriteria problem is solved by means of the multiobjective cuckoo search, which is based on an archiving technique and the crowding distance metric. Simulation results are promising when the methodology is applied to identify a robot arm given solely input and output data.

Keywords: System Identification; Multiobjective optimization; Cuckoo Search; Multiobjective Cuckoo Search; Neural Networks; Radial Basis Functions; Robotics.

## 1. INTRODUCTION

Motivated by their relatively simple architecture and property of universal approximation [Park and Sandberg, 1991], Radial Basis Function Neural Networks (RBFNNs) have been applied to a wide range of problems such as time-series forecast [Shen et al., 2011], identification of nonlinear systems [Gan et al., 2012, Ko, 2012] and control of dynamical systems [Coelho et al., 2010].

Generally the adjusted parameters of a RBFNNs are the number of neurons in the hidden layer, the position of the centers and widths of the Radial Basis Functions (RBF) and the output weights. Setting the complexity of a RBFNNs to solve an arbitrary approximation problem, though a fundamental question, is still an unsettled issue. Having defined the number of neurons on the hidden layer, the training strategy most widely used is divided in a two-stage procedure, namely (i) location of the centers through unsupervised learning and (ii) supervised learning to define the output weights. Step (i) may be performed by randomly picking points from the training set [Broomhead and Lowe, 1988] or by clustering data [Moody and Darken, 1989]. Stage (ii) may be formulated as a regression problem and thus solved by a least-mean-square such as the Penrose-Moore pseudoinverse. Though the two-stage procedure is relatively simple to implement, a RBFNNs assumes its most general form when all the centers positions, widths and output weights are obtained through supervised learning [Haykin, 2009].

Several different supervised training procedures have been proposed for RBFNNs. While some adjust solely a fixed number of parameters, others propose rules to increase or delete hidden nodes. In [Barreto et al., 2006] the authors propose a hybrid algorithm with least squares and genetic algorithms, for improving respectively local and global search for the RBFNNs parameters. Based on the neuron activity, in [Qiao and Han, 2012] the architecture of a RBFNNs is defined jointly with the estimation of the parameters. The orthogonal least square algorithm is used combined with the Bayesian information criterion, in the work of [Zhou et al., 2011], to select an appropriate number of neurons in the RBFNNs hidden layer. In [Gan et al., 2012], the authors propose for time series modeling and systems identification a hybrid methodology for improving the global-local search potential in RBFNNs training by mixing evolutionary with gradient based algorithms, where the former is run by a number of iterations and the later refines the solution at the end. Particle swarm optimization is used to perform supervised fuzzy clustering of the centers during RBFNNs design in [Tsekouras and Tsimikas, 2013].

On the other hand, in many cases as in dynamic system identification for process control, not always the inputs of the model are known a priori. On this particular, the work of [Gomm and Yu, 2000] proposes a methodology to select the lags on the input and outputs of the system in the case of NN models, by the identification of local linear models in the operation range space. In [Loghmanian et al., 2012] was proposed the optimization, through the Non-Dominated Sorting Genetic Algorithm version II, of both the accuracy of the multi-layer perceptron NN and its model structure. Based on the accuracy of (N)ARX polynomial models and their model validity, in [Chen et al., 2007] the authors

Copyright © 2014 IFAC

introduce a new fitness function optimized through genetic algorithms based on the model validity tests which defines not only the parameters of the models, but also the set of regressors of both input and output of the system.

Metaheuristic optimization techniques such as genetic algorithms, particle swarm optimization, ant colony optimization and firefly algorithm are interesting approaches to proceed the training of a NN as an optimization problem, as seen in some of the previously cited examples. Among them, the Cuckoo Search (CS) proposed by Yang and Deb [2010] has recently shown good results when compared to other techniques [Civicioglu and Besdok, 2013]. It is based in the brood parasitism behavior of cuckoos (which lay their eggs in communal nests in order to improve their chance of reproduction) together with their flight representation by Lévy distribution based random walks. As a relatively new metaheuristic technique, it has not been widely explored on the task of multiobjective optimization except on a few works [Coelho et al., 2013, Yang and Deb, 2013, Syberfeldt, 2014]. In the present work, we extend the original CS algorithm to Multiobjective Cuckoo Search (MOCS) through the use of an external archive and the crowding distance factor [Deb et al., 2000], as in [Coelho et al., 2013].

As mentioned before, the inputs of a model in the case of systems identification may not be available a priori. Moreover, the complexity of the RBFNNs is generally set manually in a tedious trial and error procedure. In the present work, we establish a training procedure which obtains all the neural network parameters and architecture – e.g. the inputs of the RBFNNs, the number of neurons in the hidden layer, the centers, widths of the RBFs and the output weights of the RBFNNs. To do so, we present an encoding scheme which includes in the optimization procedure the definition of the parameters and the architecture choice of the RBFNNs, as well as its inputs based on the system's input and output regressors. The resulting multiobjective optimization problem is solved by MOCS, showing good results when applied to real data from a robotic arm, encouraging future research endeavors. The main contribution of the present work is thus the definition of RBFNNs models for system identification based on the accuracy, model validity and regularization through an multiobjective problem formulation solved by MOCS.

The remainder of this paper is organized as follows. The Multiobjective Cuckoo Search procedure is explained in Section 2. In Section 3 we introduce the mathematical formulation of the RBFNNs models. The methodology proposed for nonlinear system identification through MOCS and RBFNNs models is detailed in Section 4. The simulation results when applying the methodology to the identification of a robot arm are given in Section 5. The conclusions and future research directions are stated in Section 6.

## 2. MULTIOBJECTIVE OPTIMIZATION FUNDAMENTALS AND CUCKOO SEARCH

In the present section, we introduce the MOCS algorithm. Section 2.1 state basic concepts used in the paper regarding Multiobjective Optimization (MO). Then, in Section 2.2, the original CS algorithm is presented. The multiobjective adaptation for CS is introduced in Section 2.3.

### *2.1 MO Fundamentals*

A general unconstrained MO problem containing a number of objectives to be minimized without loss of generality may be formulated as the minimization of $f_q(x), q = 1, 2, \ldots, Q$ where[1] $x = [x^{(1)}, x^{(2)}, \ldots, x^{(n_x)}]^T$ denote the vector of $n_x$ decision variables bounded in the interval $x^{L^{(i)}} \leq x^{(i)} \leq x^{U^{(i)}}, \forall i = \{1, 2, \ldots, n\}$ [Deb, 2001].

In the case of multiobjective optimization, all $Q$ objective functions should be optimized concomitantly. Solutions therefore cannot be compared is terms of optimality solely by examining one objective function. The concept of multiobjective optimality is introduced on the basis of dominance. Say one want to compare a solution $x_a$ with a solution $x_b$. According to the concept of dominance, a solution $x_a$ dominates a solution $x_b$, denoted by $x_a \prec x_b$, iff $f_q(x_a) \leq f_q(x_b), \forall q \in \{1, 2, \ldots, Q\}$. Possibly the objective functions are conflicting, what means that diminishing one increases another. The set of Pareto solutions (nondominated solutions) may also be termed as Pareto front. The set of solutions which represent the optimal for a multiobjective problem is said as the true Pareto set or true Pareto front.

There are in general two main goals when designing multiobjective optimization algorithms to deliver an estimated Pareto front [Deb, 2001]: (i) approximate the true Pareto front as close as possible and (ii) diversify the approximated Pareto front. Approximating the true Pareto front is related to convergence. Diversifying the approximated Pareto front provides the decision maker a set of solutions which represents trade-off among the objective functions.

The result of a multiobjective optimization procedure is an approximate set of nondominated solutions. Once the problem may require the designer to choose one solution among the Pareto set, it is possible to define metrics to proceed this choice such that the solution represents a compromise among the solutions. In this paper we utilize the minimum harmonic mean of the normalized objective functions in order to choose the best compromise solution among the approximated Pareto set.

### *2.2 Cuckoo Search*

Cuckoo Search (CS) has been recently proposed by Yang and Deb [2010] and has shown good results when compared to other nature inspired metaheuristic techniques for the resolution of optimization problems such as particle swarm optimization, differential evolution and artificial bee colony [Civicioglu and Besdok, 2013].

Cuckoos have a peculiar reproduction strategy – some cuckoo's species produce eggs and put in communal nests, while possibly removing other species's eggs. Other cuckoo's species lay eggs in the nests of other host birds – even from other species –, performing brood parasitism. Whenever a host bird detects that an egg does not belong

[1] In general the $n$-th component of a generic vector $v$ is denoted as $v^{(n)}$.

to it, it will remove those eggs from its nest or abandon the nest in order to look for a new site for it.

On the other hand, whenever looking for food animals search the environment in a random fashion. It is possible to approximate the path tracked by an animal by a random walk, for the following step depends on the current location and the probability of moving in a certain direction. Which direction will be followed can be set according to a probability that may be modeled mathematically. Studies have shown that the behavior of the flight of some animals and insects has some correspondence with the Lévy probability distribution [Yang and Deb, 2010].

The aforementioned behaviors have given researchers the idea to propose CS, a bio-inspired heuristic optimization technique. It is constituted of three simple steps [Yang and Deb, 2010]: (i) each cuckoo lays one egg at each time and puts it in a nest chosen randomly; (ii) the nest with the best eggs (solutions) proceeds to the next iteration; and (iii) the quantity of host nets is set beforehand and a host bird can discover the egg of an intruder with a given probability $p_a \in [0, 1]$ – if the host bird finds an alien egg, it will leave its nest and build a new one in a completely new location.

Let $x_i(t)$ denote the $i$-th possible solution at $t$-th iteration. Then

$$x_i(t+1) = x_i(t) \oplus \alpha \cdot \text{Lévy}(\lambda) \tag{1}$$

states the equation of the Lévy flight, where the operator $\oplus$ denotes entry-wise multiplications, $\alpha > 0$ represents the step size of the search [Yang and Deb, 2010]. The Lévy flights are essentially random walks with aleatory steps taken from a Lévy distribution

$$\text{Lévy} \sim u = t^{-\lambda}, 1 < \lambda \leq 3. \tag{2}$$

Being so, the flight of a cuckoo intrinsically represents a random walk process which follows a distribution according to a power-law with heavy tail [Yang and Deb, 2010].

The project parameters of the CS algorithm are $n$ (number of cuckoos, the size of the solutions which are performed the algorithm operators at each iteration), $p_a$ (probability of a given cuckoo leaving the nest to build one elsewhere) and $\alpha$ (the step size of the Lévy flight).

### *2.3 Multiobjective Cuckoo Search*

As reported previously in the literature, the CS algorithm – originally designed to cope with single objective optimization problems – may be adapted to a more general framework in order to solve multiobjective problems and deliver a good approximation of the Pareto front [Coelho et al., 2013, Yang and Deb, 2013].

In the present work, we adopt an archiving strategy as in [Coelho et al., 2013] to adapt the original CS algorithm to solve multiobjective optimization problems. After getting the cuckoos by Lévy flights, the new solution is evaluated and if it dominates the old one the cuckoo's position is updated in a rather greedy procedure. After getting the cuckoos, the archive is updated and the nondominated solutions are kept. The randomized search is performed by leaving the worst nests with probability $p_a$ and the nondominated check is performed once again in the new positions. The best cuckoo is chosen by sorting the archive with the Crowding Distance Factor (CDF) [Deb et al., 2000] so as to guarantee that the Pareto front approximated is also diversified and the solutions exploring not densely populated regions of the objective space are given preference. The MOCS has thus one more project parameter with respect to CS: $n_a$, which represents the size of the archive which contains the nondominated solutions.

## 3. RADIAL BASIS FUNCTION NEURAL NETWORKS

The current section gives the mathematical formulation of RBFNNs. They are typically constituted of three layers, namely the input, hidden and output layers. The input layer connects sensorially the network with the environment with source nodes to the hidden layer. The neurons in the the hidden layer have radial basis activation functions. This layer is responsible for the mapping from a nonlinear to a linear space. In the output layer, it is performed the linear weighted sum of the activations obtained in the hidden layer.

Mathematically a RBFNNs can be expressed in the following terms

$$\hat{y}(t) = \sum_{m=1}^{M} w^{(m)} \phi(r(t), c_m, \sigma_m), \tag{3}$$

where $M \in \mathbb{N}^+$ is the number of neurons in the hidden layer, $\hat{y}(t) \in \mathbb{R}$ and $r(t) \in \mathbb{R}^{n_r}$ are respectively the network predicted output and the input vector at a given instant $t$; $c_m \in \mathbb{R}^{n_r}$ and $\sigma_m \in \mathbb{R}^+$ are respectively the center and the width of the $m$-th hidden node of the neural network. The output weights are given by the vector $w \in \mathbb{R}^M$. The function $\phi(\cdot)$ is restricted to the radial basis function class. Examples of radial basis functions are the thin-plate-spline, multiquadratic, inverse multiquadratic and Gaussian functions. The activation function used more frequently in most applications (and considered in the present paper) is the Gaussian RBF

$$\phi(r, c, \sigma) = \exp\left(-\frac{||r - c||^2}{2\sigma^2}\right). \tag{4}$$

Note that we restrict to the single output case. It is possible, however, to design a set of neural networks as in (3) to extend to the multiple output case.

All three layers of RBFNNs have different roles. The input layer is responsible for connecting the network with the environment and is composed of source nodes. The second layer, which is the only hidden layer of the RBFNNs, applies a nonlinear transformation from the input to the hidden space. This nonlinear transformation is followed by a linear one, from the hidden to the output space [Haykin, 2009].

## 4. NONLINEAR SYSTEM IDENTIFICATION METHODOLOGY

The methodology for nonlinear system identification through multiobjective optimization for RBFNNs models is detailed in the present section. The procedure of defining a model and its parameters is stated as a multiobjective problem, where the error, regularization and model validations coefficients are optimized.

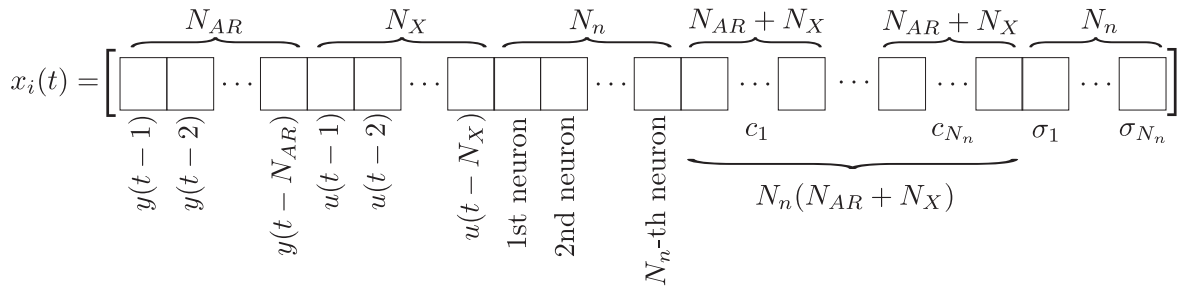


Fig. 1. Individual encoding scheme. Note that the first $N_{AR}+N_X+N_n$ elements of the individual are binary, while the rest are real numbers. The binary elements indicate the presence of the lagged outputs, inputs and neurons in the correspondent model.

Let us define some useful notation used in the current development. The one step ahead prediction error (or residual) is denoted as

$$\varepsilon(t) = y(t) - \hat{y}(t). \tag{5}$$

The normalized error for the one step ahead prediction is then given by [Chen et al., 2007]

$$e = \left( \frac{\sum_{i=1}^{N} \varepsilon(t)^2}{\sum_{i=1}^{N} y(t)^2} \right)^{1/2}, \tag{6}$$

where $N$ is the total number of samples of input and output data. The correlation function coefficients between two sequences $\{a\}$ and $\{b\}$ is given by [Billings et al., 1992]

$$\phi_{ab}(\tau) = \frac{\sum_{t=1}^{N-\tau} \left[a(t) - \bar{a}\right] \left[b(t+\tau) - \bar{b}\right]}{\left[\sum_{t=1}^{N} [a(t) - \bar{a}]^2 \sum_{t=1}^{N} [b(t) - \bar{b}]^2\right]^{1/2}}. \tag{7}$$

where the upper bar denotes the mean value of the the sequence and $-1 \leq \phi_{ab}(\tau) \leq 1$. We define the mean of the sum of the squared correlation function coefficients between two sequences $\{a\}$ and $\{b\}$ as

$$\rho_{ab} = \frac{\sum_{\tau=-\tau_{max}}^{\tau_{max}} \left[\phi_{ab}(\tau)^2\right]}{2\tau_{max}+1}, \tag{8}$$

where $\tau_{max}$ is the maximum number of lags admitted in (7). The multiple correlation coefficients are defined as [Schaible et al., 1997]

$$R^2 = 1 - \frac{\sum \varepsilon(t)^2}{\sum (y(t) - \bar{y})^2}, \tag{9}$$

For the training and validation phases we denote the multiple correlation coefficient respectively as $R_t^2$ and $R_v^2$.

The design of a RBFNNs architecture involves the definition of the number of hidden neurons and the input signals. This kind of neural network may be trained through the definition of the linear weights $w^{(m)}$, the radial basis functions centers $c_m$ and spreads $\sigma_m$. The RBFNNs assumes it most general form if its parameters are defined in a supervised way [Haykin, 2009]. In the present paper we formulate the supervised training procedure of the RBFNNs as a multiobjective optimization procedure, in order to define both the RBFNNs parameters and architecture. The vector of decision variables of the optimization procedure encodes both the architecture (inputs and number of neurons present in the hidden layer of the RBFNNs) and the project parameters. Figure 1 illustrates the solution encoding adopted. In this figure, $N_{AR}$ and $N_X$ are respectively the maximum number of lags in the output and inputs, while $N_n$ represents the maximum number of neurons allowed in the RBFNNs architecture. The first $N_{AR}$ elements of the solution vector represents the presence (1) or absence (0) of the correspondent lagged output $y(t)$ in the input of the RBFNNs. Similarly, the following $N_X$ elements give the presence/absence of the given delayed input $u(t)$. The next $N_n$ elements denote the presence or absence of each candidate neuron in the network. The following parameters are real and represent the centers and spreads of each active neuron of the RBFNNs.

Following the representation given in Fig. 1, all the parameters for the RBFNNs architecture and parameters are given, except for the output weights $w^{(m)}$. These parameters may be simply defined by a least-squares procedure as the Penrose-Moore pseudoinverse [Haykin, 2009].

In the work of Chen et al. [2007], the authors propose the use of correlation coefficients in the procedure for obtaining the parameters of NARX models. As noted in [Billings et al., 1992], the following five conditions should hold to check the validity of a neural network model: (i) $\phi_{\varepsilon\varepsilon}(\tau) = \delta(\tau)$, (ii) $\phi_{u\varepsilon}(\tau) = 0, \forall\tau$, (iii) $\phi_{(u^2)'\varepsilon}(\tau) = 0, \forall\tau$, (iv) $\phi_{(u^2)'\varepsilon^2}(\tau) = 0, \forall\tau$ and (v) $\phi_{\varepsilon(\varepsilon u)}(\tau) = 0, \tau \geq 0$, with $(u^2)'(k) = u^2(k) - \bar{u^2}$ and $(\varepsilon u)(k) = \varepsilon(k+1)u(k+1)$. Being so, in similar lines as in [Chen et al., 2007], the first objective function to be minimized during the training procedure is defined as

$$f_1(x) = e + \rho_{\varepsilon\varepsilon} + \rho_{u\varepsilon} + \rho_{(u^2)'\varepsilon} + \rho_{(u^2)'\varepsilon^2} + \rho_{\varepsilon(\varepsilon u)}. \tag{10}$$

The first term of the objective function aims at minimizing the error, while the rest focuses on the adequacy of the model through correlation tests.

The second objective function will guarantee that the model not only fits the data well, but also generalizes the dynamic system aiming at regularization:

$$f_2(x) = \frac{1}{M} \sum_{m=1}^{M} (w^{(m)})^2. \tag{11}$$

While using this objective function, the candidate RBFNNs which have smaller output weights will be given preference. That tend to lead to RBFNNs which avoid overfitting for having smoother outputs.

### *4.1 Penalty procedure*

During the search procedure the solution shown in the scheme in Fig. 1 may present undesired response. We penalize solutions which are not feasible from the point

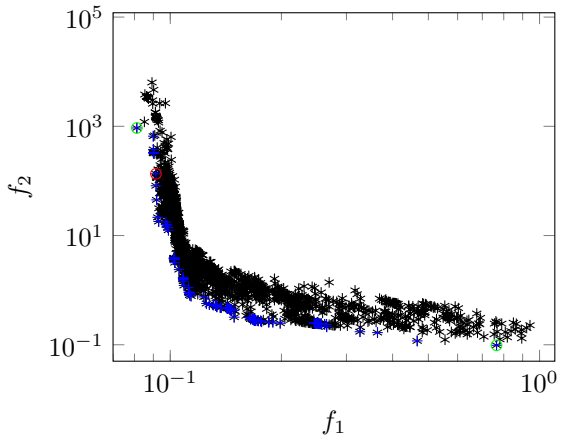


Fig. 2. Pareto optimal solutions (black stars) obtained at each of the 30 runs and nondominated solutions from this set (blue stars). The best compromise solution is shown with a red circle, while the minimum for each objective function is denoted with green circles.

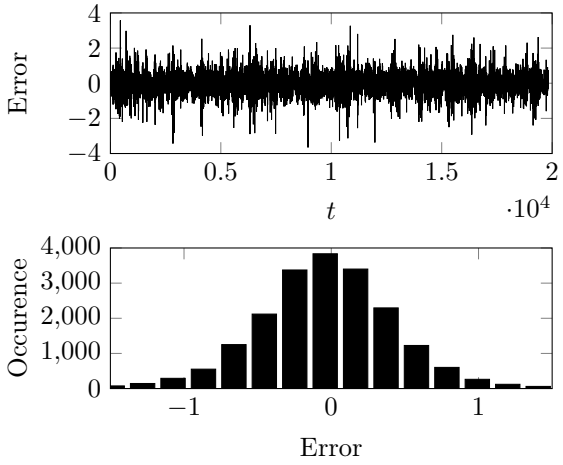


Fig. 3. Best compromise solution error (upper) and error histogram (lower) for both training and validation phases.

of view of the architecture, that is, solutions that present no neurons in the hidden layer or no regressors in the output and input. RBFNNs obtained whose performance present values for the multiple correlation coefficient in the training phase smaller than 0.9 are also penalized [Schaible et al., 1997].

## 5. SIMULATION RESULTS

The present section shows the results obtained when solving the multiobjective problem formulated with the objective functions as in (10) and (11) with the individual encoding scheme illustrated in Fig. 1 by MOCS introduced in the subsection 2.3.

We test the methodology introduced in Section 4 with real data acquired from an experimental industrial robot [Wernholt and Gunnarsson, 2006]. In this paper, the authors propose a methodology for nonlinear grey-box identification, in order to obtain the parameters of the arm including flexibility in the gearbox and in the arm

Table 1. Solutions found by MOCS and its results: number of neurons in the hidden layer, neural network inputs and multiple correlation coefficients in the training and validation phases.

| Solution | $M$ | RBFNNs input | $R_t^2$ | $R_v^2$ |
|---|---|---|---|---|
| $x_{f_1}$ | 6 | $r(t) = \begin{bmatrix} y(t-1) \\ y(t-2) \\ y(t-3) \\ y(t-4) \\ u(t-1) \end{bmatrix}$ | 0.9995 | 0.9994 |
| $x_{f_2}$ | 5 | $r(t) = \begin{bmatrix} y(t-1) \\ y(t-3) \\ u(t-5) \end{bmatrix}$ | 0.9566 | 0.9313 |
| $x_c$ | 6 | $r(t) = \begin{bmatrix} y(t-1) \\ y(t-3) \\ u(t-1) \\ u(t-2) \\ u(t-3) \\ u(t-4) \end{bmatrix}$ | 0.9989 | 0.9989 |

structure and friction in the system. The focus in the present work, however, is to obtain a black-box model for the system, assuming that no a priori information about its dynamics is available. The system output $y(t)$ is the velocity at the end effector, while the input $u(t)$ is the torque at the joint motor. The dataset used contains 19,838 samples, divided into training and validation phases (50% split).

The project parameters related to the RBFNNs architecture utilized were $N_n = 10$, $N_{AR} = 4$ and $N_X = 5$. With respect to MOCS, the project parameters were set to $n = 50$, $p_a = 0.25$, $\alpha = 0.01$ and $n_a = 50$. We used $\tau_{max} = 20$ to calculate the correlation coefficients. The first two parameters were set using the guidelines provided in [Yang and Deb, 2013]. The parameter $n_a$ should not be set arbitrarily high in order to maintain solely representative solutions in the archive. The stopping criteria is 500 iterations of the algorithm. The multiobjective optimization problem is solved by MOCS for the minimization of the objective functions $f_1$ and $f_2$ in 30 independent runs with different initial conditions.

Figure 2 illustrates with black stars all solutions found by running MOCS 30 times with different initial conditions. In this plot, the resulting nondominated solutions when comparing the results of all 30 runs are represented by blue stars. From this set, the objective functions for $x_{f_1}$ (minimum $f_1(x)$) and $x_{f_2}$ (minimum $f_2(x)$)are shown by green circles and the $x_c$ (best compromise solution) is represented by a red circle. The results concerning the solution $x_c$ are shown in Fig. 3, which gives the plot of the error and its histogram (for both training and validation phases). Table 1 shows the number of neurons ($M$), lagged inputs and outputs selected as the RBFNNs input ($r(t)$) and the multiple correlation coefficients $R_t^2$ and $R_v^2$.

The resulting Pareto front given in Fig. 2 shows that the optimization procedure was able to find a diverse nondominated set with 115 solutions by running 30 times. Comparing all 30 runs, there was 77.17 solutions in average in the final Pareto front, with a maximum of 118 and a minimum of 38 nondominated solutions. The best compromise solution RBFNNs results detailed in Table 1

show the network accuracy and capacity of generalization, illustrated by the values of $R_t^2$ and $R_v^2$. We can also see the prediction errors and their good statistical properties from Fig. 3.

## 6. CONCLUSION

The present paper evaluated a methodology for incorporating both the complexity and input selection for RBFNNs models in its training procedure. Towards this end, we proposed a solution encoding scheme to formulate the RBFNNs training procedure as a multiobjective optimization problem and solved it by MOCS. The MOCS algorithm has been shown effective to solve the problem. It is important to mention that the methodology assumes that no a priori information about the system dynamics (input and output regressors and model complexity) is available.

Results showed the validity of the proposed methodology and encourage further research, which will aim at (i) application to more complex and multivariable systems; (ii) extension of the formulation to time-series forecasting; (iii) comparison with other metaheuristics; (iv) generalization of the proposed encoding scheme to other types of neural networks and to other classes of models, e.g nonlinear autoregressive moving-average with exogenous inputs.

## ACKNOWLEDGEMENTS

This study was partially supported by the Brazilian National Council of Scientific and Technological Development (CNPq) under Grants 479764/2013-1 and 307150/2012-7/PQ and by CAPES (Brazilian research agency) through a PROSUP scholarship.

## REFERENCES

A. M. S. Barreto, H. J. C. Barbosa, and N. F. F. Ebecken. GOLS—genetic orthogonal least squares algorithm for training RBF networks. *Neurocomputing*, 69(16):2041–2064, 2006.

S. A. Billings, H. B. Jamaluddin, and S. Chen. Properties of neural networks with applications to modelling non-linear dynamical systems. *Int. J. Control*, 55(1): 193–224, 1992.

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

Q. Chen, K. Worden, P. Peng, and A.Y.T. Leung. Genetic algorithm with an improved fitness function for (N)ARX modelling. *Mech. Syst. Signal Pr.*, 21(2):994–1007, 2007.

P. Civicioglu and E. Besdok. A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. *Artif. Intell. Rev.*, 39(4):315–346, 2013.

L. S. Coelho, M. W. Pessôa, R. S. Rodrigues, and A. A. R. Coelho. Model-free adaptive control design using evolutionary-neural compensator. *Expert Syst. Appl.*, 37(1):499–508, 2010.

L. S. Coelho, F. A. Guerra, N. J. Batistela, and J. V. Leite. Multiobjective cuckoo search algorithm based on Duffing's oscillator applied to Jiles-Atherton vector hysteresis parameters estimation. *IEEE T. Magn.*, 49 (5):1745–1748, 2013.

K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE T. Evolut. Comput.*, 6(2):182–197, 2000.

M. Gan, H. Peng, and L. Chen. A global–local optimization approach to parameter estimation of RBF-type models. *Inform. Sciences*, 197:144–160, 2012.

J. B. Gomm and D. L. Yu. Order and delay selection for neural network modelling by identification of linearized models. *Int. J. Syst. Sci.*, 31(10):1273–1283, 2000.

S. S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 2009.

C.-N. Ko. Identification of non-linear systems using radial basis function neural networks with time-varying learning algorithm. *IET Signal Process.*, 6(2):91–98, 2012.

S. M. R. Loghmanian, H. Jamaluddin, R. Ahmad, R. Yusof, and M. Khalid. Structure optimization of neural network for dynamic system modeling using multiobjective genetic algorithm. *Neural Comput. Appl.*, 21 (6):1281–1295, 2012.

J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Comput.*, 1(2): 281–294, 1989.

J. Park and I. W. Sandberg. Universal approximation using Radial-Basis-Function networks. *Neural Comput.*, 3(2):246–257, 1991.

J.-F. Qiao and H.-G. Han. Identification and modeling of nonlinear dynamical systems using a novel self-organizing RBF-based approach. *Automatica*, 48(8): 1729–1734, 2012.

B. Schaible, H. Xie, and Y.-C. Lee. Fuzzy logic models for ranking process effects. *IEEE T. Fuzzy Syst.*, 5(4): 545–556, 1997.

W. Shen, X. Guo, C. Wu, and D. Wu. Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. *Knowl.-Based Syst.*, 24(3):378–385, 2011.

A. Syberfeldt. Multi-objective optimization of a real-world manufacturing process using cuckoo search. In X.-S. Yang, editor, *Cuckoo Search and Firefly Algorithm*, volume 516 of *Studies in Computational Intelligence*, pages 179–193. Springer, 2014.

G. E. Tsekouras and J. Tsimikas. On training RBF neural networks using input–output fuzzy clustering and particle swarm optimization. *Fuzzy Sets and Systems*, 221:65–89, 2013.

E. Wernholt and S. Gunnarsson. Nonlinear identification of a physically parameterized robot model. In *14th IFAC Symposium on System Identification*, pages 143–148, Newcastle, Australia, 2006.

X.-S. Yang and S. Deb. Engineering optimisation by cuckoo search. *Int. J. Math. Model. Numer. Optim.*, 1(4):330–343, 2010.

X.-S. Yang and S. Deb. Multiobjective cuckoo search for design optimization. *Comput. Oper. Res.*, 40(6): 1616–1624, 2013.

P. Zhou, D. Li, H. Wu, and F. Cheng. The automatic model selection and variable kernel width for RBF neural networks. *Neurocomputing*, 74(17):3628–3637, 2011.