Evolutionary Multiobjective Neural Network Models Identification: Evolving Task-Optimised Models

Pedro M. Ferreira and António E. Ruano

Abstract. In the system identification context, neural networks are *black-box* models, meaning that both their parameters and structure need to be determined from data. Their identification is often done iteratively in an *ad-hoc* fashion focusing the first aspect. Frequently the selection of inputs, model structure, and model order are underlooked subjects by practitioners, because the number of possibilities is commonly huge, thus leaving the designer at the hands of the *curse of dimensionality*. Moreover, the design criteria may include multiple conflicting objectives, which gives to the model identification problem a multiobjective combinatorial optimisation character. *Evolutionary multiobjective optimisation algorithms* are particularly well suited to address this problem because they can evolve optimised model structures that meet pre-specified design criteria in acceptable computing time. In this article the subject is reviewed, the authors present their approach to the problem in the context of identifying neural network models for time-series prediction and for classification purposes, and two application case studies are described, one in each of these fields.

1 Introduction

In most practical applications of Artificial Neural Networks (ANN), they are used to perform a non-linear mapping between an input space, $\mathbf{X}$, and an output space, $\mathbf{y}$, in order to model complex relationships between these or to detect patterns in

Pedro M. Ferreira
Algarve Science & Technology Park, University of Algarve,
Campus de Gambelas - Pav. A5, 8005-139 Faro, Portugal
e-mail: pfrazao@ualg.pt

António E. Ruano · Pedro M. Ferreira
Centre for Intelligent Systems, University of Algarve - FCT,
Campus de Gambelas, 8005-139 Faro, Portugal
e-mail: aruano@ualg.pt

A. Ruano et al. (Eds.): New Advances in Intelligent Signal Processing, SCI 372, pp. 21–53.
springerlink.com

input-output data. These functionalities correspond mostly to function approximation problems in the context of static or dynamic models identification, or decision problems in the contexts of pattern matching and classification. The non-linear mapping function illustrated in Fig. 1 is given by:

$$\hat{y}_k = g(\mathbf{x}_k, \mathbf{w}) \tag{1}$$

Usually, given a data set $\mathbf{D} = (\mathbf{X}, \mathbf{y})$ composed of N input-output pairs, the network

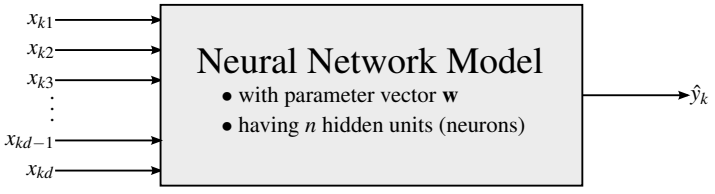

Fig. 1 Illustration of a general mapping $\hat{y}_k = g(\mathbf{x}_k, \mathbf{w})$

parameter vector $\mathbf{w}$ is computed in order to minimise the sum-of-squares of the mapping error, i.e.,

$$\varepsilon = \mathbf{e}^T \mathbf{e} \tag{2}$$

where,

$$\begin{aligned} &\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}, && e_k = y_k - \hat{y}_k, \\ &\hat{\mathbf{y}} = g(\mathbf{X}, \mathbf{w}), && \hat{y}_k = g(\mathbf{x}_k, \mathbf{w}), \\ &\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T, && \mathbf{y} = [y_1, y_2, \ldots, y_N]^T, \\ &\mathbf{x}_k = [x_{k1}, x_{k2}, \ldots, x_{kd}]. \end{aligned} \tag{3}$$

In many applications the set of d input features, x_{ki}, needs to be selected from a larger set, $\mathbf{F}$, often having a dimension significantly larger than a prescribed maximum input vector dimension, d_M. Assuming $\mathbf{F}$ has q features it may be specified as,

$$\begin{aligned} &\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_q], \\ &\mathbf{f}_l = [f_{1l}, f_{2l}, \ldots, f_{Nl}]^T. \end{aligned} \tag{4}$$

Then, the input data set $\mathbf{X}$ is constructed by selecting d columns from $\mathbf{F}$ such that,

$$\begin{aligned} \mathbf{x}_k &= [x_{k1}, x_{k2}, \ldots, x_{kd}] = \\ &= \left[f_{k\lambda_1}, f_{k\lambda_2}, \ldots, f_{k\lambda_d}\right], \end{aligned} \tag{5}$$

where the λ_j are indices to the columns of $\mathbf{F}$.

Also, depending on the type of ANN to be employed, the number n of hidden units (artificial neurons or simply neurons) must be specified. Once d input features

are selected and the number of neurons, n, is specified, the ANN parameter vector, $\mathbf{w}$, is computed by means of a suitable training algorithm.

By taking into account these three aspects of the ANN model identification process, the problem addressed in this article may be generally stated as follows:

> Considering the application at hands, select d ($d_m < d \leq d_M$) input features from the set $\mathbf{F}$, a suitable number of neurons n ($n_m < n \leq n_M$), and compute the ANN parameter vector $\mathbf{w}$, such that the *best* ANN mapping, given in (1), is obtained.

It is already clear that the ANN design problem may be separated in two distinct sub-problems, each reflecting different aspects of the design:

ANN parameters	relates to the network parameters. Includes their computation by means of a training algorithm.
ANN structure	relates to the network topology. Includes the selection of suitable inputs and an appropriate number of neurons;

Many techniques have been proposed to solve both sub-problems, either separately or jointly, some failing to capture their distinct nature, therefore not fully exploiting existing approaches that are considered more appropriate. The first is a non-linear parameter optimization problem, to which non-linear gradient-based methods have proven to be superior. The second is a combinatorial optimisation problem that, as will be shown, needs to be addressed from a multiobjective optimisation perspective.

In Sect. 2 a more precise definition of the problem statement will be given and the approach followed successfully by the authors in a number of applications will be presented. From these, two were selected and are described in Sect. 3, one in the field of time-series modelling and forecasting, the other in the area of classification problems. The results from the application of the methodology to the selected ANN design problems will be presented and discussed in Sects. 3.1.2 and 3.2.2, respectively for each design problem. Finally, some concluding remarks will be made in Sect. 4.

2 Methodology

The problem statement presented in the previous section is of a general nature leaving open two vague notions that need to be elaborated in order to provide a formal problem definition. On one hand the concept of ”best ANN mapping” requires the definition of *best*. On the other, the sentence ”considering the application at hands” implies that the problem will be solved by taking the application into account. In fact the two notions are related as it seems appropriate to define what is a ”*best* ANN, considering the application at hands”.

2.1 Problem Definition

In order to define a "*best* ANN, considering the application at hands", one or more quality measures are required so that any two different ANN solutions may be compared and a decision can be reached on which is best. The problem decomposition given in Sect. 1 suggests the existence of quality measures for each sub-problem and gives clear hints on how to choose them:

ANN parameters	The quality measures should reflect how well did the training stage performed and how good is the mapping obtained by the parameters computed.
ANN structure	The quality measures should tell how fit is the ANN structure for the application at hands.

This breakdown in the nature of the quality measures allows the definition of a two component quality vector as,

$$\begin{aligned} \mu\left(\mathbf{F},\Lambda,n,\mathbf{w}\right) &= \left[\mu^p,\mu^s\right], \\ \mu^p &= \left[\mu_1^p,\mu_2^p,\ldots,\mu_u^p\right], \\ \mu^s &= \left[\mu_1^s,\mu_2^s,\ldots,\mu_v^s\right], \\ \Lambda &= \left[\lambda_1,\lambda_2,\ldots,\lambda_d\right], \end{aligned} \tag{6}$$

where μ^p and μ^s contain u and v quality measures related to each of the sub-problems, Λ is the vector of indices to the columns of $\mathbf{F}$ that defines the input features considered, and the superscripts p and s denote quality measures related to the ANN training stage and to the ANN fitness for the specific application, respectively. The dependence on $\mathbf{F}$, Λ, n, and $\mathbf{w}$, has been made explicit only for μ for easiness of reading.

Assuming that the quality measures in $\mu\left(\mathbf{F},\Lambda,n,\mathbf{w}\right)$ are well defined quantities specifying objective functions that should be minimised in order to obtain the "best ANN for the application at hands", the problem statement given in Sect 1 may now be formally defined as:

Select $d \in [d_m, d_M]$ input features from $\mathbf{F}$, $n \in [n_m, n_M]$ neurons, and compute $\mathbf{w}$, such that $\mu\left(\mathbf{F},\Lambda,n,\mathbf{w}\right)$ is minimised. Formally,

$$\begin{aligned} &\min_{\Lambda,n,\mathbf{w}} \mu\left(\mathbf{F},\Lambda,n,\mathbf{w}\right), \text{ given:} \\ &\left(\mathbf{F},\mathbf{y}\right), \\ &d \in [d_m, d_M], \\ &n \in [n_m, n_M]. \end{aligned} \tag{7}$$

Given the definition of $\mu\left(\mathbf{F},\Lambda,n,\mathbf{w}\right)$ (simply μ in the following), it is likely that some objective functions are conflicting, e.g. in μ^P Eq. 2 may be minimised and in

μ^s there could be an objective to minimise the complexity of the ANN, expressing the goals of improving performance while decreasing the network size. Therefore the search problem defined in (7) is a combinatorial multiobjective optimisation problem which does not have a single solution minimising all components of μ simultaneously. Instead, the solution is the set of Pareto points in search space (or design space) that define the Pareto front in the space of objectives. This means that the ANN model designer has to select one particular ANN by examining trade-offs in the objectives of the Pareto front.

Searching exhaustively over the search space defined by $(\mathbf{F}, [d_m, d_M], [n_m, n_M])$ is the preferred solution as it allows finding the true Pareto front, but this is normally unfeasible in useful time due to the complexity of evaluating μ and to the size of the search space. Although trial and error may provide an approach to guide the search, the number of possibilities is often enormous and it may result in the execution of many trials without obtaining acceptable objective values in μ. Moreover, the results from the trials may easily misguide the designer into some poor local minima as the relation between search space and objective space is unknown.

Although a good number of techniques have been proposed over the years to deal with multiobjective problems it was more recently that the potential of Evolutionary algorithms (EAs) to approximate the Pareto front was recognised, generating a research area now known as evolutionary multiobjective optimisation (EMO). Multiobjective evolutionary algorithms (MOEAs) have proven to be robust and efficient when dealing with problems with multiple conflicting objectives and with very large and complex search spaces, therefore they are employed here to solve the ANN structure search problem. A review about the EMO field and MOEAs is beyond the scope of this article, the interested reader can find detailed descriptions in textbooks, e.g. [10], and excellent overviews on [48, 7, 8].

The application of EAs to the design of ANN models has been addressed by many researchers, with variations on the aspects of ANN design that are considered. Distinct formulations employ EAs in order to optimise/select: the number of neurons and network parameters [32, 5, 3, 6, 47, 29]; both the topology and parameters [28, 25]; the complete topology [27, 2]; or, only the network inputs [34]. A discussion considering the different possible formulations may be found in [4]. The approach herein presented follows previous work [21, 36] in the context of polynomial models identification. It has been applied by the authors in the contexts of time-series modelling and prediction [43, 16, 17, 41, 14, 37, 18, 42], and classification [9, 12].

2.2 Multiobjective Evolutionary Algorithms

MOEAs are one class of EAs that benefit from a set of procedures and operators inspired on the process of natural evolution and on the notion of survival of the fittest, in order to perform a population based search for the Pareto set of solutions of a given multiobjective problem. The solution candidates are called *individuals* and their set is referred to as the *population*. One run of a MOEA starts with an

initial population of individuals, the initial *generation*, which are then evaluated and manipulated to compute the population of individuals composing the next generation. Hopefully, after a sufficient number of generations the population has evolved achieving a satisfactory approximation to the Pareto front.

The operation of most MOEAs follows the flow illustrated in Fig. 2 where the main procedures and operators are shown. At each iteration the population is

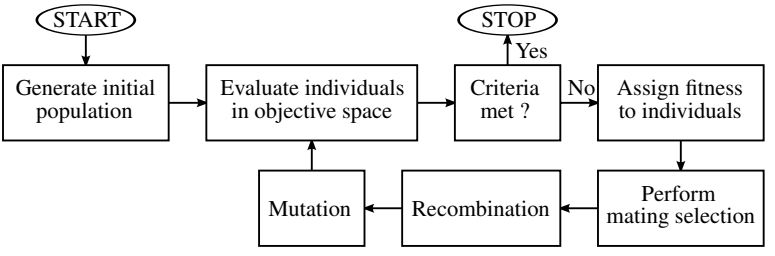

Fig. 2 Typical flow of operation of most MOEAs

evaluated for the objectives specified in μ and a check is made to ascertain if the design criteria was met. If this is the case the MOEA stops and the designer obtains the individuals that form the current approximation to the Pareto front, otherwise the algorithm proceeds. In this case each individual in the population is assigned a fitness value and based on this fitness the individuals are mated. Afterwards each mated pair will produce two offspring by the application of the recombination operator, thus forming the next generation. Finally the mutation operator is applied to each children before repeating the whole process.

2.2.1 Individual Representation Each individual in the MOEA population must be specified by a representation, the *chromosome*, encoding the topology of an ANN. Most frequently, *feed-forward* ANNs are employed in modelling, prediction and classification problems, usually having one or two hidden layers of neurons. In the following the general class of feed-forward ANNs having one hidden layer of neurons is considered. As will be shown, if two or more layers are used only slight changes are required in the chromosome and in the mutation operator.

The topology of the ANN architectures just mentioned may be completely specified by the number of neurons n and by the indices Λ to the columns of $\mathbf{F}$, defining the input features to be employed. Therefore the chromosome is a string of integers, the first representing the number of neurons and the remaining representing the subset of input terms taken from $\mathbf{F}$. The chromosome definition is shown in Fig. 3. The multiobjective optimisation problem defined in (7) states that the number of inputs d is required to be in the range $[d_m, d_M]$. This corresponds to a variable length chromosome having at least d_m input terms. The first component corresponds to the number of neurons, those highlighted by a light grey background represent the minimum number of inputs, and the remaining are a variable number of input terms up

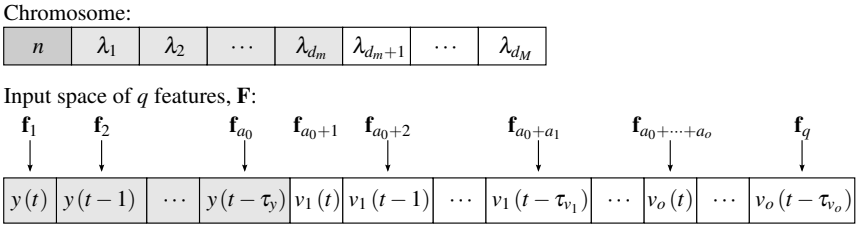

Fig. 3 Chromosome and input space lookup table

to (in total) d_M. The λ_j values are the indices of the features $\mathbf{f}_l$ in the columns of $\mathbf{F}$. In cases where the ANN acts as a predictor the input-output structure is, in the most general form, given by a non-linear autoregressive (NAR) with exogenous inputs (NARX),

$$\begin{aligned} y(t+1) = g\,(&y(t), y(t-1), \cdots, y(t-\tau_y), \\ &v_1(t), v_1(t-1), \cdots, v_1(t-\tau_{v_1}), \\ &\cdots, \\ &v_o(t), v_o(t-1), \cdots, v_o(t-\tau_{v_o}))\,, \end{aligned} \tag{8}$$

where y is the output and v_1 to v_o are o exogenous inputs. In such cases $\mathbf{F}$ is composed of a_0 output delayed terms with a maximum lag of τ_y and a_i input terms for each exogenous variable v_i, each having τ_{v_i} as maximum lag. The correspondence between the features in the columns of $\mathbf{F}$ and the inputs of a NARX model is depicted in the lower part of Fig. 3, where the inputs corresponding to delayed output values are highlighted by a light grey background.

It should be noted that the chromosome would require a small change if the ANN considered had multiple hidden layers. In this situation as many additional components as the number of additional hidden layers would be inserted at the beginning of the chromosome, in order to encode the number of neurons in the various layers.

2.2.2 MOEAs Procedures and Operators Once evaluated in objective space each individual is assigned a scalar value, the *fitness*, that should reflect that individual's quality. The fitness assignment strategy is one of the distinguishing characteristics of existing MOEAs, thus is usually dependant on the MOEA used in practice. In general these strategies are based on different principles and belong to one of three classes: aggregation based; criterion based; and Pareto based strategies. For more detailed discussions on these strategies, the reader should consult the literature on the MOEA being used or one of the references given above about MOEAs.

The mating procedure uses the population fitness information in order to create a *mating pool* with pairs of individuals that will be combined to form the basis of the next generation population. It is commonly implemented as a sampling procedure where the individuals having higher fitness have increased chance of getting multiple copies in the mating pool, and those with lower fitness have little or no chance

of getting there. The result is that the fittest individuals have a higher probability of breeding as opposed to the worse individuals that are unlikely to influence the new generation.

With a given probability, the *crossover probability*, every pair of individuals in the mating pool produces two offspring by exchanging part of their chromosomes. This is accomplished by the recombination operator, whose operation is illustrated in Fig. 4 by splitting the procedure in two steps. First, the chromosomes are reordered, secondly, parts of the chromosomes are exchanged. Reordering is also

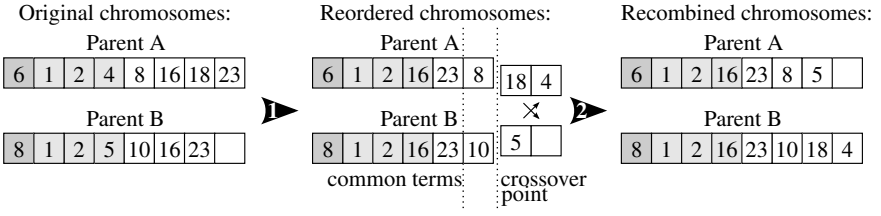

Fig. 4 Crossover recombination operator

accomplished in two steps: common terms in the chromosomes are swapped to the left-most positions, then the remaining terms are shuffled. This way the common terms in the chromosomes are isolated in a way that makes them unavailable for the exchange. A point is then randomly chosen, the *crossover point*, and the elements to its right are exchanged. This procedure, known as *full identity preserving crossover* [21, 24], guarantees offspring with no duplicate terms.

Mutation is applied to the new population generated after recombination, independently in two parts of the chromosome. The number(s) of neurons in the hidden layer(s) of the ANN are mutated with a given probability by adding or subtracting one neuron to the existing quantity. Care must be taken in order to guarantee the boundary conditions $n_m \leq n \leq n_M$. The input terms are mutated with a given probability by one of three operations: replacement, addition or deletion. First, each term is tested and is either deleted or replaced by another term from the set of those outside the chromosome. Deletion only occurs if the chromosome has more terms than the minimum specified, d_m. After this, if the chromosome is not full, one term may be added by selecting it from the set of those outside the chromosome.

After completing the operations described above the MOEA flow proceeds to the evaluation step and the cycle repeats itself for the new population of individuals. Therefore, some criterion is required to stop the MOEA execution. The most simple approach consists in stopping the execution after a predefined number of generations. Other options include testing the objectives and design criteria and stop the execution if a satisfactory individual is found, or checking the population for stagnation. The latter option may be accomplished, for instance, by specifying a maximum number of consecutive generations during which no change is observed in the Pareto front approximation.

2.3 Model Design Cycle

Globally, the ANN structure optimisation problem can be viewed as sequence of actions undertaken by the model designer, which should be repeated until pre-specified design goals are achieved. These actions can be grouped into three major categories: problem definition, solution(s) generation and analysis of results. In the context of this identification framework, the procedure is executed as depicted in Fig. 5. In

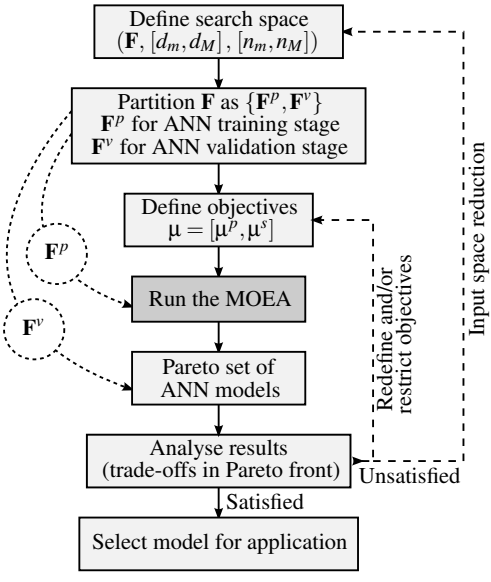

Fig. 5 Model design cycle

summary, the problem definition is carried out by choosing a number of hypothetically relevant input features to construct $\mathbf{F}$, by specifying the minimum and maximum size of the chromosome input terms string, and by defining the range allowed for the number of neurons of the ANNs. In the case of a NARX predictive model identification, the specification of $\mathbf{F}$ corresponds to the selection of input variables and the corresponding lags considered. This stage affects the size of the search space. The input search space is then partitioned into two data sets, $\mathbf{F}^p$ and $\mathbf{F}^v$, the first intended for the ANN parameter training procedure, the second to validate the results obtained by the Pareto set of individuals. The validation step serves the purpose of detecting any bias that may have occurred towards the $\mathbf{F}^p$ data set during the MOEA model structure optimisation.

Another aspect to be defined is the set of objectives and goals to be attained. The objectives specified in μ^s play an important role in the adequacy of the models obtained to the application problem being considered. Therefore they should be designed to express the quality of an individual in the context of the final application. Specifying μ affects the quantity, quality and class of the resulting solutions.

When the analysis of the solutions provided by the MOEA requires the process to be repeated, the problem definition steps should be revised. In this case, two major actions can be carried out: input space redefinition by removing or adding one or more features (variables and lagged input terms in the case of modelling problems), and improving the trade-off surface coverage by changing objectives or redefining goals. This process may be advantageous as usually the output of one run allows reducing the number of input terms (and possibly variables for modelling problems) by eliminating those not present in the resulting population. Also, it usually becomes possible to narrow the range for the number of neurons in face of the results obtained in one run. This results in a smaller search space in a subsequent run of the MOEA, possibly achieving a faster convergence and better approximation of the Pareto front. This cycle of actions can be iterated until a refined set of satisfactory solutions is obtained.

2.4 ANN Parameter Training

In Sects. 1 and 2.1, the ANN identification problem has been decomposed in two sub-problems, the first one, related to the optimisation of the network parameters, being usually treated as a non-linear optimisation problem. It is clear that the training procedure is most often dependant on the specific ANN being employed, although some procedures may easily be adapted to various kinds of ANNs. The class of feed-forward ANNs include, among others, radial basis function (RBF) networks, multi-layer perceptrons (MLPs), B-spline networks, wavelet networks, and some types of neuro-fuzzy networks. Importantly, a common topology of these architectures share the property of *parameter separation* , i.e., they can be regarded as a non-linear/linear topology because one or more hidden layers of non-linear neurons are followed by a linear combination of neuron outputs to produce the network overall result. It is commonly accepted that gradient-based algorithms, in particular the Levenberg-Marquardt (LM) algorithm [31], outperforms other parameter training methods, and it has been shown that methods exploiting the separability of parameters [39, 40, 13] achieve increased accuracy and convergence rates.

The two example ANN identification problems that will be introduced in Sect. 3 employ RBF neural networks (NNs) and the LM algorithm in the minimisation of a modified training criterion that exploits the separability of parameters as found in RBF NNs. For this reason an outline of the training procedure is given in the following sub-sections.

2.4.1 Training Criterion For simplicity, but without loss of generality, feed-forward ANNs having one hidden layer of neurons are considered. These may be well represented by the expression,

$$\hat{y}(\mathbf{x}_k, \mathbf{w}) = \alpha_0 + \sum_{i=1}^{n} \alpha_i \varphi_i(\mathbf{x}_k, \beta_i) \,, \tag{9}$$

where $\mathbf{w} = [\alpha, \beta]^T$ is the model parameter vector, $\alpha = [\alpha_0, \alpha_1, \cdots, \alpha_n]$ is the vector of scalar linear parameters, and $\beta = [\beta_1, \cdots, \beta_n]$ is composed of n β_i vectors of non-linear parameters, each one associated with one neuron. For a given set of input patterns $\mathbf{X}$, training the NN corresponds to finding the values of $\mathbf{w}$ such that (10) is minimised:

$$\Omega(\mathbf{X}, \mathbf{w}) = \frac{1}{2} \|\mathbf{y} - \hat{\mathbf{y}}(\mathbf{X}, \mathbf{w})\|^2 \tag{10}$$

The $\frac{1}{2}$ factor is used for convenience considering the training algorithm to be employed. As the model output is a linear combination of the neuron activation functions output, (10) may be written as,

$$\Omega(\mathbf{X}, \mathbf{w}) = \frac{1}{2} \|\mathbf{y} - \phi(\mathbf{X}, \beta)\alpha\|^2 \tag{11}$$

where, omitting the dependence of φ on β,

$$\phi(\mathbf{X}, \beta) = [\varphi(\mathbf{x}_1)\ \varphi(\mathbf{x}_2)\ \cdots\ \varphi(\mathbf{x}_N)]^T .$$

By computing the optimal value α^* of the linear parameters α with respect to the non-linear parameters β, as a least-squares solution,

$$\alpha^* = \phi^+(\mathbf{X}, \beta)\mathbf{y}, \tag{12}$$

where "+" denotes a pseudo-inverse operation, and by replacing (12) in (11), the training criterion to compute the non-linear parameters is obtained:

$$\Psi(\mathbf{X}, \beta) = \frac{1}{2} \left\|\mathbf{y} - \phi(\mathbf{X}, \beta)\phi^+(\mathbf{X}, \beta)\mathbf{y}\right\|^2 . \tag{13}$$

This criterion is independent of the linear parameters α and explicitly incorporates the finding that, whatever values the non-linear parameters β take, the α^* parameters employed are the optimal ones. Moreover, it reflects the non-linear/linear parameters structure of the feed-forward ANN model in (9), by separating their computation. This way it becomes possible to iteratively minimise (13) to find β^*, corresponding to searching for the best non-linear mapping, and then solve (12) using β^* to obtain the complete optimal parameter vector $\mathbf{w}^*$. The modified criterion enables the usage of appropriate methods to compute each type of parameters in the minimisation of a single explicit criterion. It lowers the dimensionality of the problem and usually achieves increased convergence rate.

2.4.2 Training Algorithm Various training algorithms can be employed to minimise (10) or (13). First-order gradient algorithms (known for MLPs as the *back-propagation* algorithm) or second-order methods, such as quasi-Newton, Gauss-Newton or LM can be employed as training algorithm. For non-linear least-squares problems the LM algorithm is recognised as the best method, as it exploits the sum-of-squares characteristic of the problem [38].

Denoting the standard (10) or modified (13) training criteria in iteration k by $\Omega(\mathbf{w}_k)$ (omitting the dependence of Ω on $\mathbf{X}$), a search direction $\mathbf{p}_k$ in parameter space is computed such that $\Omega(\mathbf{w}_k + \mathbf{p}_k) < \Omega(\mathbf{w}_k)$. This method is said to be of the restricted step type because it attempts to define a neighbourhood of $\mathbf{w}_k$ in which a quadratic function agrees with $\Omega(\mathbf{w}_k + \mathbf{p}_k)$ in some sense. The step $\mathbf{p}_k$ is restricted by the region of validity of the quadratic function which is obtained by formulating in terms of $\mathbf{p}_k$ a truncated Taylor series expansion of $\Omega(\mathbf{w}_k + \mathbf{p}_k)$. Then, it may be shown that $\mathbf{p}_k$ can be obtained by solving the following system [31]:

$$\left(\mathbf{J}_k^T \mathbf{J}_k + \upsilon_k \mathbf{I}\right) \mathbf{p}_k = -\mathbf{g}_k \,. \tag{14}$$

$\mathbf{g}_k$ and $\mathbf{J}_k$ are, respectively, the gradient and Jacobean matrix of $\Omega(\mathbf{w}_k)$, $\upsilon_k \geq 0$ is a scalar controlling the magnitude and direction of $\mathbf{p}_k$. By recalling (3), the gradient may easily be obtained as,

$$\begin{aligned} \mathbf{g}_k &= \frac{\partial \Omega(\mathbf{w}_k)}{\partial \mathbf{w}_k} = \\ &= -\mathbf{J}_k^T \mathbf{e}_k \,, \end{aligned} \tag{15}$$

where the Jacobean matrix has the form:

$$\mathbf{J}_k = \begin{pmatrix} \frac{\partial y_1}{\partial w_1} & \cdots & \frac{\partial y_1}{\partial w_l} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_N}{\partial w_1} & \cdots & \frac{\partial y_N}{\partial w_l} \end{pmatrix} \tag{16}$$

The advantage of the LM algorithm is that in every iteration the value of υ is adapted in order to provide a step direction more close to the Gauss-Newton or gradient-descent methods. When $\upsilon \to 0$ the step direction approaches that of the Gauss-Newton method, when $\upsilon \to \infty$ it approaches the gradient-descent direction. Many variations of Marquardt's algorithm have been proposed concerning the rules governing the adaptation of υ. The original method [31] or a similar one [20, 11] should suit most applications.

2.4.3 RBF Network The RBF ANN is formulated by (9) where the i^{th} basis function or neuron, $\varphi_i(\mathbf{x}_k, \beta_i)$, is usually a Gaussian, a multiquadric, or an inverse multiquadric function. In most cases the Gaussian is employed:

$$\varphi_i(\mathbf{x}_k, \beta_i) = \exp\left(-\frac{1}{2\sigma_i^2} \|\mathbf{x}_k - \mathbf{c}_i\|^2\right) \tag{17}$$

In this case $\beta_i = [\mathbf{c}_i \; \sigma_i]$ is the non-linear parameter vector where $\mathbf{c}_i$ is a point in input space, the centre of the Gaussian function, and σ_i is the corresponding spread. The outputs of all neurons are then linearly combined (recall Eq. 9) to produce the network output.

At the first iteration of the training algorithm the model parameters have to be initialised. Common approaches consist in selecting $\mathbf{c}_i$ randomly from the input pat-

terns or from the input variables range of values. An alternative is to take advantage of clustering algorithms in order to spread the centres in distinct regions of the input feature space. The σ_i parameters may be chosen randomly, or, for instance using the simple rule [26, p. 299],

$$\sigma_i = \frac{z^{max}}{\sqrt{2n}}, \tag{18}$$

where z^{max} is the maximum Euclidean distance among the initial centres $\mathbf{c}_i$, and n is the number of neurons. Once the vectors of non-linear parameters, β_i, are initialized, (12) may be employed to determine the initial linear parameters, α_i.

In order to employ the LM algorithm to optimise the RBF ANN parameter vector, the error criterion must be defined as well as the derivatives required. If the standard formulation (10) is used, the three derivatives required to compute $\mathbf{J}$ are:

$$\begin{aligned}
\frac{\partial y}{\partial \mathbf{c}_i} &= \varphi_i(\mathbf{x}) \frac{\alpha_i}{\sigma_i^2} (\mathbf{x} - \mathbf{c}_i)^T, \\
\frac{\partial y}{\partial \sigma_i} &= \varphi_i(\mathbf{x}) \frac{\alpha_i}{\sigma_i^3} \|\mathbf{x} - \mathbf{c}_i\|^2, \\
\frac{\partial y}{\partial \alpha_i} &= \varphi_i(\mathbf{x}).
\end{aligned} \tag{19}$$

For the modified criterion (13) alternative Jacobean matrices are available. It has been shown [39] that a simple and efficient solution consists in using the two first lines of (19), where the α_i are replaced by their optimal values as computed in (12). Remarkably, the use of this Jacobian matrix implies that each iteration of the LM method minimising (13) is computationally cheaper than minimising (10).

2.4.4 Stopping the Training Algorithm As most ANN training algorithms are iterative, some criteria is required to stop the training procedure after a certain number of iterations. Whichever method is employed, it should prevent the algorithm to *overtrain* the network parameters. Overtraining is a "phenomenon" likely to occur when using iterative training algorithms, characterised by a distinct behaviour of the error criterion when computed on two, distinct, data sets. On the data set employed to estimate the model parameters (training data set), the error criterion decreases with the number of iterations usually reaching a plateau where improvements become negligible. If a second data set is used to test the model at every iteration, then the error criterion taken on the testing data set decreases to a certain iteration and starts to increase in subsequent iterations. Beyond this point overtraining occurred because too many iterations were executed and the model became biased by the training examples, thus loosing the capability to generalise properly when presented with new input patterns. The methods of regularisation and *early stopping* are probably the most common to avoid overtraining. The first is a technique based on extending the error criterion with a penalty term, therefore numerically changing the training method, the second is a data driven cross-validation approach that may

be viewed as an implicit regularisation method. Interesting in-depth reading about the early stopping and overtraining subjects may be found in [1, 45].

In practice overtraining is avoided by stopping the training algorithm before reaching the absolute minimum of the training criterion. The method of early stopping, requires splitting the training input space $\mathbf{F}^p$ (see Fig. 5, Sect. 2.3) into two data sets, the first, $\mathbf{F}^t$, to estimate the model parameters, called training data set, the second, $\mathbf{F}^g$, to assess the model generalisation capability, called generalisation data set. In the model design cycle presented earlier, by using a given MOEA individual chromosome, $\mathbf{F}$ is indexed by the input terms in the chromosome. After indexing, the input data set is denoted by $\mathbf{X}$ and the input-output data set by $\mathbf{D} = (\mathbf{X}, \mathbf{y})$. Consequently the training, generalisation, and validation data sets, for a given ANN, will be defined as,

$$\begin{aligned} \mathbf{D}^t &= \left(\mathbf{X}^t, \mathbf{y}^t\right) , \\ \mathbf{D}^g &= \left(\mathbf{X}^g, \mathbf{y}^g\right) , \\ \mathbf{D}^v &= \left(\mathbf{X}^v, \mathbf{y}^v\right) , \end{aligned}$$

for training, generalisation testing, and validation, respectively. Recall that $\mathbf{D}^v$ is meant to validate the MOEA model optimisation globally, to avoid bias towards $\mathbf{D}^t$ and $\mathbf{D}^g$ in the final model selection. The proportions of points from $\mathbf{D}$ that compose $\mathbf{D}^t$ and $\mathbf{D}^g$ are often selected in an ad hoc fashion, usually by means of trial and error. A statistically validated principled way of selecting that proportion may be found in [1].

By denoting the error criterion computed on the testing and generalisation data sets at iteration k by $\Omega\left(\mathbf{D}^t, \mathbf{w}_k^*\right)$ and $\Omega\left(\mathbf{D}^g, \mathbf{w}_k^*\right)$, the early stopping method consists in selecting the model parameters corresponding to the iteration where $\Omega\left(\mathbf{D}^g, \mathbf{w}_k^*\right)$ ceased to decrease (assuming $\Omega\left(\mathbf{D}^t, \mathbf{w}_k^*\right) \leq \Omega\left(\mathbf{D}^t, \mathbf{w}_{k-1}^*\right)$). In practice the inflection point on the $\Omega\left(\mathbf{D}^g, \mathbf{w}_k^*\right)$ curve must not be identified locally by a rule of the type $\Omega\left(\mathbf{D}^g, \mathbf{w}_k^*\right) > \Omega\left(\mathbf{D}^g, \mathbf{w}_{k-1}^*\right)$ as this method would be sensitive to small variations that are still occurring in a more global descending trend. An alternative is to define k^{max} as the maximum number of iterations to execute and then find the global minimum of $\Omega\left(\mathbf{D}^g, \mathbf{w}_k^*\right)$. Formally, assuming monotonically decreasing $\Omega\left(\mathbf{D}^t, \mathbf{w}_k^*\right)$, this may be written as,

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \left\{\Omega\left(\mathbf{D}^g, \mathbf{w}_k^*\right)\right\}_{k=1}^{k^{max}} , \tag{20}$$

where k^{max} should be large enough to include the global minimum of $\Omega\left(\mathbf{D}^g, \mathbf{w}_k^*\right)$. When this method is used to stop the training algorithm used in a MOEA model identification experiment, $\Omega\left(\mathbf{D}^t, \mathbf{w}^*\right)$ and $\Omega\left(\mathbf{D}^g, \mathbf{w}^*\right)$ are commonly included in μ^p expressing the goal of identifying models achieving a good data fitting and good generalisation capability.

If early-stopping is not possible, an alternative is to employ a set of termination criteria that is commonly used in unconstrained optimisation [23, p. 306]. Let θ_k, a measure of absolute accuracy, be defined as,

$$\theta_k = \tau^{\Omega} \times \left(1 + \left|\Omega\left(\mathbf{D}^t, \mathbf{w}_k^*\right)\right|\right) , \tag{21}$$

where τ^{Ω} is a measure of the desired number of correct digits in the objective function. The optimisation stops when all the following conditions are met:

$$\Omega\left(\mathbf{D}^t, \mathbf{w}_{k-1}^*\right) - \Omega\left(\mathbf{D}^t, \mathbf{w}_k^*\right) < \theta_k \tag{22}$$

$$\left\|\mathbf{w}_{k-1}^* - \mathbf{w}_k^*\right\| < \sqrt{\tau^{\Omega}}\left(1 + \|\mathbf{w}_k^*\|\right) \tag{23}$$

$$\|\mathbf{g}_k\| \leq \sqrt[3]{\tau^{\Omega}}\left(1 + \left|\Omega\left(\mathbf{D}^t, \mathbf{w}_k^*\right)\right|\right) \tag{24}$$

The two first conditions test the convergence of the model parameters. The reasoning behind the use of two conditions is that for ill-conditioned problems, $\Omega\left(\mathbf{D}^t, \mathbf{w}_k^*\right)$ may be a good approximation of the global minimum (22), but $\mathbf{w}_k^*$ may be far from the optimum and the algorithm may still be making large adjustments to $\mathbf{w}^*$ (23). The third condition reflects the necessity that the gradient should be near zero if $\Omega\left(\mathbf{D}^t, \mathbf{w}_k^*\right)$ is close to the optimum. This method achieves a certain level of regularisation, implicitly related to the parameter τ^{Ω}, and does not require the $\mathbf{D}^g$ data set, therefore lowering the number of function evaluations required by the inclusion of $\Omega\left(\mathbf{D}^g, \mathbf{w}_k^*\right)$ in the objective space of the model identification problem. The disadvantage is that the resulting number of training iterations might not be enough to adequately converge the model parameters, or it might be in excess and provoke overtraining.

3 Example Model Identification Problems

To exemplify the use of the methodologies presented in the sections above, two ANN model identification problems that the authors have been involved with will be discussed. The first deals with the prediction of the Portuguese electricity consumption profile within an horizon of 48 hours, the second is related to the estimation of cloudiness from ground-based all-sky hemispherical digital images.

3.1 Electricity Consumption Prediction

The Portuguese power grid company, *Rede Eléctrica Nacional* (REN), aims to employ electricity load demand (ELD) predictive models on-line in their dispatch system to identify the need of reserves to be allocated in the Iberian market. To accomplish this the evolution of ELD over a prediction horizon of at least 48 hours is required. The problem is addressed from the point of view of identifying RBF ANN one-step-ahead ELD predictive models using the framework already described. These models are iterated in a multi-step fashion in order to predict the electricity consumption profile up to the specified prediction horizon.

In previous work [19] a literature review on the ELD forecasting area was presented, demonstrating that the approach taken was relevant and ambitious as no publications were found considering simultaneously four aspects that the team is actively addressing:

Prediction scheme	To meet the requirement, one-step-ahead predictive models are iterated in a multi-step fashion in order to obtain the consumption profile up to the specified prediction horizon. Most work relies on the prediction of daily peak consumption or accumulated consumption over a certain period, and does not consider dynamics.
Model adaptation	On-line model adaptation strategies are necessary, as the models are static mappings with external dynamics and the profiles of electricity consumption vary over time.
Perturbations	One input is incorporated in the models to account for the effect of events that dramatically perturb the typical profile of load demand (the effect of week-ends, holidays and other foreseeable events).
Optimised models	The problem of model structure optimisation and selection is clearly formulated and approached by appropriate methodologies in order to meet specified design requirements.

By that time a number of exploratory identification experiments were executed which allowed refining the problem formulation (see Sect. 2.3, model design cycle). In the following, one last identification experiment is described, from which one model was selected [18]. It is currently in operation at the Portuguese power-grid company dispatch system.

3.1.1 Problem Formulation Two types of model structures were previously compared [19], the NAR and the NARX. For the latter only one exogenous input was considered, encoding the occurrence of events perturbing the daily and weekly patterns of electricity consumption. That comparison favoured the NARX approach as it consistently achieved a considerably better prediction accuracy.

Table 1 Day of the week and holiday occurrence encoding values

Day of week	Regular day	Holiday	Special
Monday	0.05	0.40	0.70
Tuesday	0.10	0.80	
Wednesday	0.15	0.50	
Thursday	0.20	1.00	
Friday	0.25	0.60	0.90
Saturday	0.30	0.30	
Sunday	0.35	0.35	

The exogenous input encoding, presented in table 1, distinguishes between the days of the week and also the occurrence and severity of holidays based on the day of their occurrence. The *regular day* column shows the coding for the days that are not

holidays. The next column presents the encoded values when there is a holiday for that day of the week, and finally, the *special* column shows the values that substitute the regular day value in two special cases: for Mondays when Tuesday is a holiday; and, for Fridays when Thursday is a holiday. Figure 6 illustrates the severity of the perturbation that a holiday causes in the electricity consumption profile. It is evident

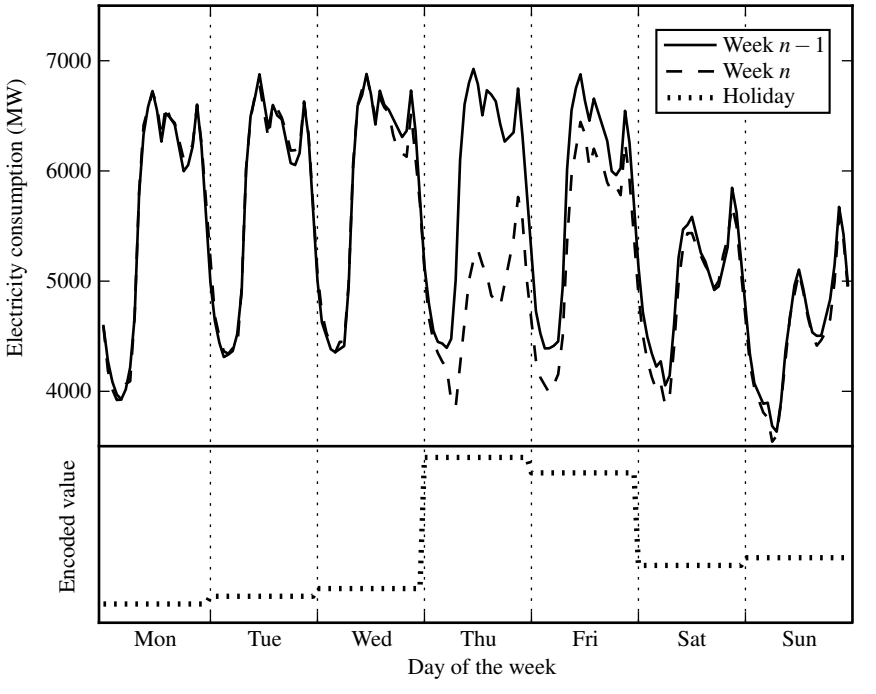

Fig. 6 ELD in two consecutive weeks. A holiday occurs in the second week.

that not only during the holiday the change is very large, but also on the following day a significant change may be observed.

The data used in the model identification experiment corresponds to the Portuguese electrical energy consumption measured at hourly intervals, for a time span starting around mid October 2007, and ranging to the end of 2008. The complete time-series is presented in Fig. 7. It was split in four data sets for model training, generalisation testing, predictive simulation, and validation. The points for each set were selected from three distinct periods of the year, delimited in Fig. 7 by two vertical dotted lines to the right of the plot. $\mathbf{D}^t$ and $\mathbf{D}^g$ are composed of 330 and 60 days of data points randomly selected from the first period. The last 50 days of 2008 were divided in two parts, the first being used as a simulation data set, $\mathbf{D}^s$, the second as the validation data set, $\mathbf{D}^v$. Taking into account the use of one input encoding the

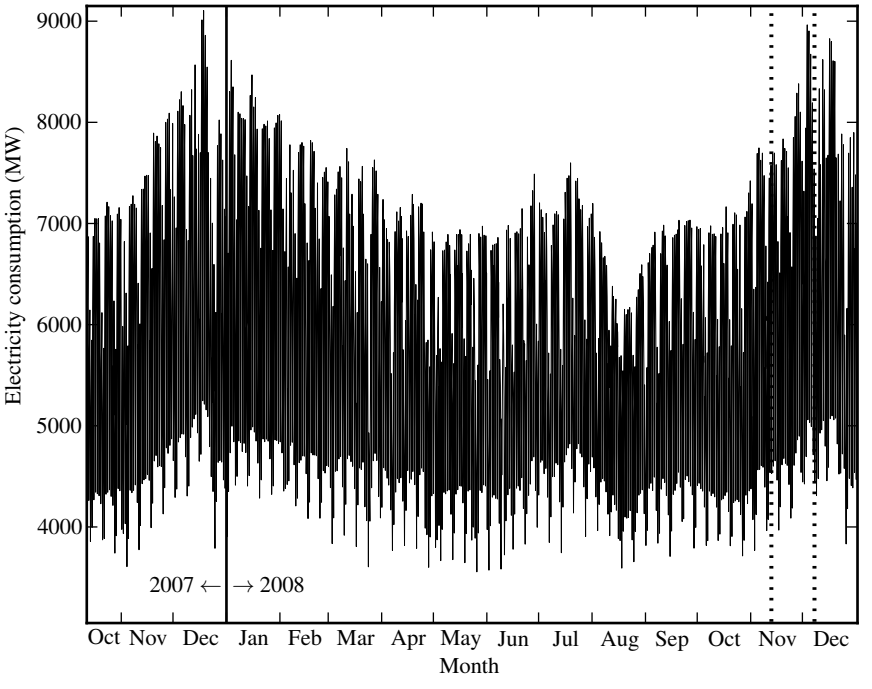

Fig. 7 ELD in Portugal for the time span considered

occurrence of weekends and holidays, care was taken to make sure that the data sets $\mathbf{D}^s$ and $\mathbf{D}^v$ included holidays.

The lookup table $\mathbf{F}$, from which the four data sets are built for each ANN by indexation using the input part of the chromosomes, is composed of 168 delayed ELD input terms plus the input encoding the occurrence and severity of holidays. The ELD input terms correspond to one week window, an interval for which the time-series exhibits a clear repetitive pattern. This pool of candidate input terms was specified by considering the results of previous experiments. Also the limits for the number of neurons and for the size of the input part of the chromosome were specified by taking previous results into account. In this case significant changes were made by doubling the maximum number of neurons, $n \in [10, 28]$, and by increasing the maximum number of input terms allowed, $d \in [2, 40]$.

The model parameters were estimated via the LM algorithm using the modified training criterion (13) as outlined in Sect. 2.4. The initial centre locations for the Gaussian activation functions were selected randomly from the input patterns in $\mathbf{D}^t$, the corresponding initial spreads, σ_i, were determined by the rule (18), and the linear parameters were initialised using (12). The early stopping method was employed to stop the training algorithm by setting $k^{max} = 200$ in (20).

In order to address the model structure selection problem, the multiobjective genetic algorithm (MOGA) [22] is employed to evolve a *preferable set* of models whose number of neurons and selected input terms optimise a number of pre-

specified goals and objectives. These, as discussed in Sect. 2.1, are specified by a two component vector of objective functions, $\mu = [\mu^p, \mu^s]$. For the first component, related to the ANNs parameter training process, two model performance objectives were considered, given by the root-mean-square (RMS) error computed on the training and generalisation testing data sets, respectively denoted by $\rho(\mathbf{D}^t)$ and $\rho(\mathbf{D}^g)$. The first one is used as a restriction because a clear positive (linear or not) relationship between the training criterion and a long-term prediction performance (to be defined below) is not guaranteed and was not observed in practice in previous experiments. One last objective was specified for the first component of μ, given by the 2-norm of the linear parameters vector, $\|\alpha\|$. It is employed as a restriction in order to guarantee good numerical properties and parameter convergence in the models, but in fact it also acts as a penalty term for the complexity of the model. Regarding μ^s, the component of μ related to the model structure selection and to the specific model application, one objective was considered expressing the final goal of the model application: the prediction of the electricity consumption profile within an horizon of 48 hours. It is computed on the basis of the long-term model prediction error taken from the multi-step model simulation over the prediction horizon *ph*. Assume that a given simulation data set, $\mathbf{D}$, has p data points and for each point the model is used to make predictions up to *ph* steps ahead. Then an error matrix is constructed,

$$\mathbf{E}(\mathbf{D}, ph) = \begin{pmatrix} e[1,1] & e[1,2] & \cdots & e[1,ph] \\ e[2,1] & e[2,2] & \cdots & e[2,ph] \\ \vdots & \vdots & \ddots & \vdots \\ e[p-ph,1] & e[p-ph,2] & \cdots & e[p-ph,ph] \end{pmatrix},$$

where $e[i,j]$ is the model prediction error taken from instant i of $\mathbf{D}$, at step j within the prediction horizon. Denoting the RMS function operating over the i^{th} column of its argument matrix by $\rho(\cdot, i)$, then the long term prediction performance measure is defined as,

$$\varepsilon(\mathbf{D}, ph) = \sum_{i=1}^{ph} \rho(\mathbf{E}(\mathbf{D}, ph), i), \tag{25}$$

which is simply the summed RMS of the columns of $\mathbf{E}$. This way the single objective in μ^p is simply given by $\varepsilon(\mathbf{D}^s, 48)$. This represented a considerable change from previous work as the prediction horizon was doubled from 24 to 48 hours.

The complete objective vector for the ELD prediction problem is therefore specified as $\mu = [\rho(\mathbf{D}^t) \quad \rho(\mathbf{D}^g) \quad \|\alpha\| \quad \varepsilon(\mathbf{D}^s, 48)]$. Table 2 summarises the objectives and their configuration as used in the MOGA ELD predictive modelling experiment. As the ANN parameters are randomly initialised, for each individual, 10 training trials were executed and the averages of $\rho(\mathbf{D}^t)$ and $\rho(\mathbf{D}^g)$ were used for evaluation purposes. This procedure decreases the likelihood of unrealistic fitness assignment in the MOEA as one good ANN structure could be poorly evaluated due to a bad choice of initial parameters. In order to decrease the computational load, $\varepsilon(\mathbf{D}^s, 48)$

Table 2 Objective space configuration for the MOGA ELD prediction problem

μ component	Objective function	Set up as
μ^p	$\rho(\mathbf{D}^t)$	restriction $< 100\ MW$
	$\rho(\mathbf{D}^g)$	to minimise
	$\|\alpha\|$	restriction < 200
μ^s	$\varepsilon(\mathbf{D}^s, 48)$	minimise

was only computed for the trial instance whose pair $\{\rho(\mathbf{D}^t), \rho(\mathbf{D}^g)\}$ is closer (in the Euclidean sense) to the averages over the 10 trials.

3.1.2 Results and Discussion Figure 8 illustrates the results obtained in the space of objectives after 50 generations of the MOGA (values in Mega Watt (MW)).At this generation the execution

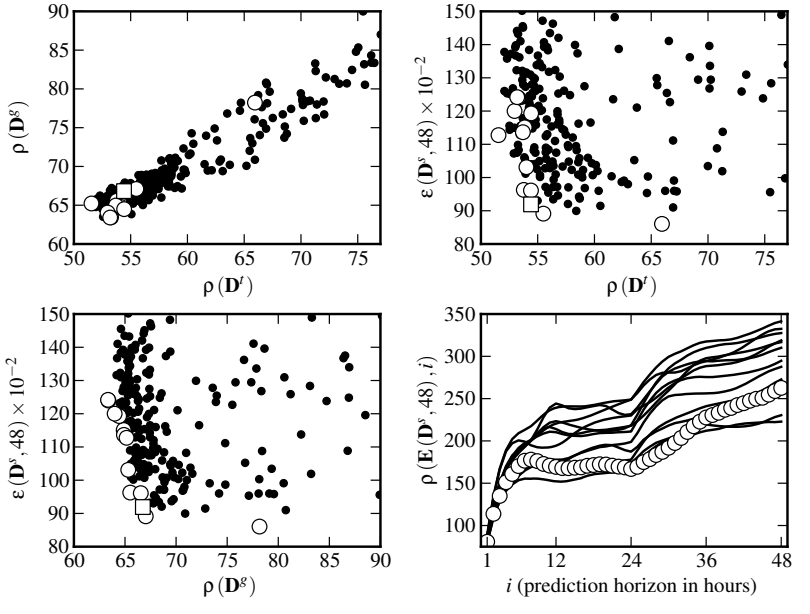

Fig. 8 Objectives of the MOGA ELD predictive model identification experiment

was stopped as the last models having lower values on $\varepsilon(\mathbf{D}^s, 48)$ entered the preferable set on generation 30, and convergence was only occurring for $\rho(\mathbf{D}^t)$ and $\rho(\mathbf{D}^g)$ with no effect on $\varepsilon(\mathbf{D}^s, 48)$. This considerably smaller number of generations, when compared to previous work, clearly shows the benefits of using averaged objective values over multiple model training trials. The three scatter plots show the results of non-dominated individuals using dark points, and the results of the 13 models in the preferable set using white circles (and one white square). The top-left plot shows a linear relation between the error criterion obtained on the training and generalisation testing data sets. The plots at the top-right and bottom-left show the relation between each error criterion and the long-term prediction error measure, where the conflict between these objectives is well demonstrated .The lower-right plot presents the evolution of $\rho(\mathbf{E}(\mathbf{D}^s, ph), i)$ with i from 1 to ph, the prediction horizon. The curve marked with white circles was obtained by the model marked using a white square on the remaining plots. The objective values are slightly better than those obtained in previous work, however it should be noted that the prediction horizon was doubled. The 13 selected models had from 24 to 28 neurons, 26 the most frequent, and from 34 to 39 input terms, 36 the most frequent. All of them included the holiday encoding input, and other 15 input terms were employed in 10 models or more. When compared to previous work, the increase in model complexity is explained by a slightly better predictive accuracy over a double size prediction horizon.

The models obtained were evaluated on the validation data set, $\mathbf{D}^v$, in order to select one for further assessment of predictive accuracy and robustness. Considering that during the MOGA execution only one out of 10 models was evaluated for $\varepsilon(\mathbf{D}^s, 48)$, for each of the models in the preferable set 10 further training trials were executed and their performance was evaluated on $\mathbf{D}^s$ and $\mathbf{D}^v$. The results are depicted in Fig. 9 where the dark markers highlight the results obtained by the chosen model structure (marked by white squares in Fig. 8), the square marker corresponding to the instance selected for further study. This was accomplished by comparing

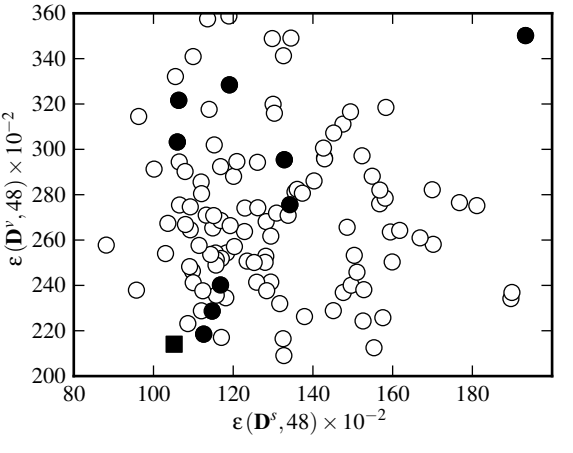

Fig. 9 Detail of the long term prediction error measure obtained on the simulation data set, $\mathbf{D}^s$, versus that obtained in the validation data set, $\mathbf{D}^v$, for the preferable set of 13 models. Ten training trials per model.

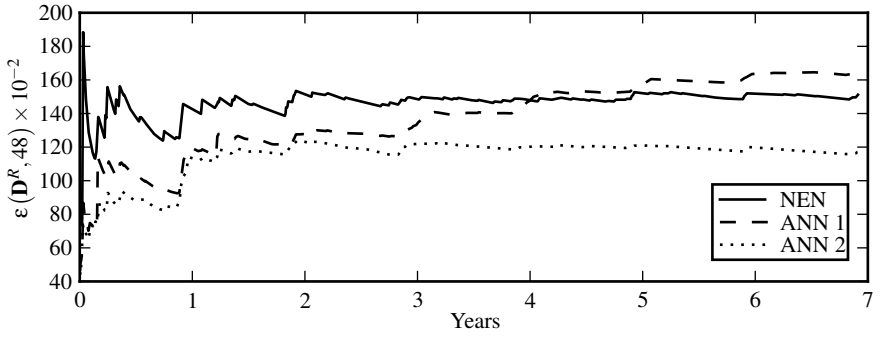

Fig. 10 Comparison between the selected ANN (ANN 1), a nearest neighbour approach (NEN), and the selected ANN with a yearly retraining (ANN 2)

the selected model structure to a nearest neighbour (NEN) predictive approach on a data set (denoted by $\mathbf{D}^R$) ranging from the beginning of 2001 to the end of 2008. A complete description of the NEN methodology is given in [18]. It was tested by varying the number of nearest neighbours employed for prediction and by using a sliding window of past 54 weeks to conduct the nearest neighbour search. The selected ANN was used to predict the ELD on the same instants (≈ 7 years) as the NEN method, being initially trained using data from the first sliding window (approximately the 2001 year). Figure 10 shows the evolution of $\varepsilon\left(\mathbf{D}^R, 48\right)$ as time increases. It may be seen that the NEN method is quite robust as the prediction performance measure converges asymptotically to a value near 150×10^{-2}. The line labelled ANN 1, corresponding to the model structure selected, shows that as time passes $\varepsilon\left(\mathbf{D}^R, 48\right)$ tends to increase at an almost constant rate, becoming higher than that of the NEN method after about 4 years of data. This is likely to happen because the ANN parameters no longer reflect with the same accuracy the underlying dynamics and trend of the ELD time series, leading to the conclusion that the ANN requires some form of adaptation to become robust. Even so it is quite remarkable that, with no parameter change, it achieves better prediction accuracy during the first four years of data (significantly better in the first three years). In order to obtain a fair comparison with the NEN method (in the sense that it uses a sliding window of information), another set of results was computed by retraining the ANN at every year interval so that its parameters are readjusted to reflect more closely the ELD data dynamics and trends. These results are labelled ANN 2 in Fig. 10, showing that the improvements are significant even though only a yearly retraining was employed. In terms of robustness this is very promising for the actual implementation of ANN ELD predictive models in the REN dispatch system, as further improvements are expectable if more elaborate model adaptation techniques are employed or a more frequent retraining is used [15]. Figure 11 shows the evolution of the ELD over the prediction horizon for the simulation instant where the retrained ANN achieved the RMS error value which was closest to the average over the complete simulation. The

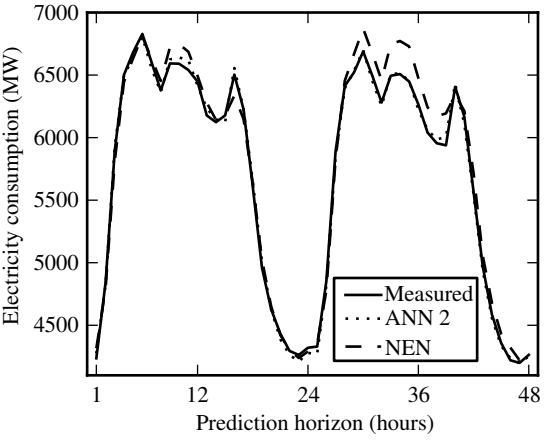

Fig. 11 Prediction horizon where the retrained model (ANN 2) RMS of error is closest to the average obtained in the complete simulation. The nearest neighbour approach prediction is shown for comparison.

prediction obtained by the NEN method is also shown for comparison. Globally the results show that the ANN is preferable to the NEN method, although at the cost of a significant increase in methodology complexity and on computational effort.

In summary, when compared to previous results, better prediction accuracy was achieved over a longer prediction horizon, and faster convergence (in number of generations) was observed in the MOGA execution.

3.2 Cloudiness Estimation

Clouds are an important phenomena strongly affecting the total incoming irradiance at a given point in the Earth surface. For a growing number of applications in diverse fields such as agriculture, forestry or energy production and management, being able to accurately estimate and predict solar radiation at a given ground location and at short time scales, is becoming an extremely important task because solar radiation strongly influences the relevant processes and energy balances. The use of ground-based all-sky (GBAS) images acquired by CCD cameras, directly with fish-eye lenses or projected on hemispherical mirrors, has been receiving growing interest by researchers from several fields (See [12] for examples and references). Regarding the use of cloudiness information extracted from GBAS images and its incorporation into solar radiation predictive modelling, our group made a first attempt in a previous work [9]. The pixel classification approach was quite different from that being presented here and there was no assessment, other than by visual inspection, on the accuracy of the cloud cover estimation. By that time no clear conclusion could be made on the benefits of using cloudiness information at the inputs of the neural network predictive solar radiation model. Typically predictive solar radiation models are identified using one-step-ahead NAR forms. Due to the autoregressive characteristic their accuracy becomes severely degraded in the presence of cloudy sky conditions, hence the need to advance to the NARX form, considering cloudiness has the exogenous input.

The motivation for this work is twofold: to improve the predictive performance of global solar radiation models operating on relatively short time scales (prediction horizons of a few hours); and, to implement these models on a cheap hardware daytime all-sky imaging prototype being developed in the laboratory. Ultimately our goals are to employ global solar radiation predictive models incorporating the effects of cloudiness in projects related to the efficient energy management in public buildings and, in the future, in projects related to solar power plants and to the prediction of electricity consumption.

3.2.1 Problem FormulationA total of 410 all-sky images were used in the model identification experiment. They were acquired using a *Total Sky Imager* (TSI) *440A* manufactured by *Yankee Environmental Systems, Inc.*, located on top of one building ($37^o02'N$, $07^o57'W$) in the University of Algarve, Faro, Portugal. The images are stored in red-green-blue (RGB) colour mode (8 bit/channel) with a dimension of 704×576 (width$\times$height). Given the location of the TSI and the time-stamp of each image, a pixel mask was computed to identify the visible sky pixels for further processing (see Fig. 13 for an example). For these, one researcher made an additional mask including all the cloud pixels according to his personal judgement. Using these masks the percent cloud cover for each image was computed using the formula,

$$C = \frac{N_c}{N_s + N_c} \times 100\,, \tag{26}$$

where N_s and N_c are the numbers of pixels masked as clear sky (class **S**) and cloud (class **C**), respectively. Figure 12 presents information about the images used, illustrating the effort made to include significant numbers of images within intervals of the cloud cover and the time of day. Additionally, for each pixel intensity scale considered and for every image, an exhaustive search was conducted to find the threshold value, t_o, minimising the cloud cover estimation error resulting from the thresholding operation.

The general approach consists in finding a threshold value, $\hat{t}$, on a given pixel intensity scale, which segments the image I pixels with coordinates (x,y) and intensity γ_{xy} into one of the classes, **S** and **C**. In this sense these are sets defined as,

$$\mathbf{S} = \left\{(x,y) \in I : \gamma_{xy} \leq \hat{t}\right\}\,,$$
$$\mathbf{C} = \left\{(x,y) \in I : \gamma_{xy} > \hat{t}\right\}\,,$$

to which N_s and N_c in (26) are the respective set cardinalities. The evaluation of thresholding methods relies on the absolute error between the cloud fraction attributed to the images and that estimated by the threshold $\hat{t}$:

$$\varepsilon = \left|C - \hat{C}_{\hat{t}}\right|\,. \tag{27}$$

Several pixel intensity scales were considered to perform the thresholding operation. From the results in [12] one, denoted *hsvR*, was selected has it consistently provided increased cloud cover estimation accuracy for various thresholding methods tested.

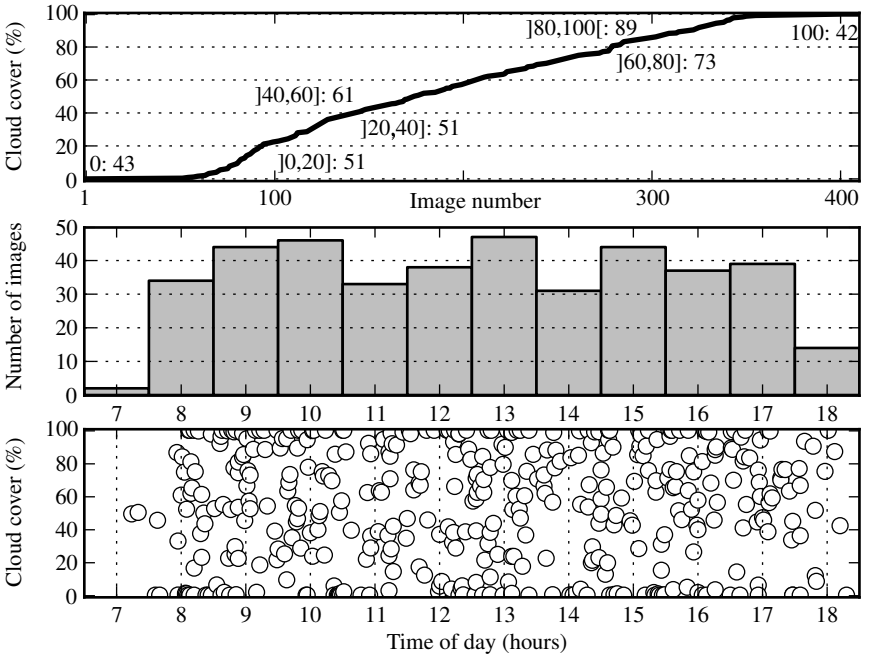

Fig. 12 Details about the 410 images used. Top: Number of images by ascending cloud cover intervals. Middle and bottom: respectively, the number of images and cloud cover distribution by the time of day.

This pixel intensity is obtained by converting the original RGB image to the *hue-saturation-value* (HSV) colour model, setting the value channel to 1 (the maximum) for all pixels, and finally converting this image back to the RGB mode. Setting an equal value on the V channel has an equalisation effect on the pixels luminosity. The maximum value was chosen because on the HSV colour model the colours become more distinguishable. The net effect on the converted RGB image is that clear sky and cloud pixels have improved contrast between them in the red channel.

The model identification problem consists in using the framework presented in Sect. 2 in order to search for a RBF ANN image segmentation model. As illustrated in Fig. 13, the output of the ANN is the threshold to be used in an image, the inputs are a set of features extracted from the masked image or transformations of it.

The set of 410 images was broken into three sub-sets: the training set, denoted by $\mathbf{D}^t$ (290 images); the testing set, $\mathbf{D}^g$ (60 images), for generalisation testing; and the validation set, $\mathbf{D}^v$ (60 images), to evaluate the ANNs after the MOEA execution. From all the images and from transformations of them, a total of 69 features were extracted from distinct pixel intensity scales: first, from the original RGB image the HSV and hue-saturation-lightness (HSL) images were obtained; secondly, on the HSV and HSL images, the V and L channels were set to 1 and 0.5, respectively, and these transformed images were converted back to RGB mode, thus generating

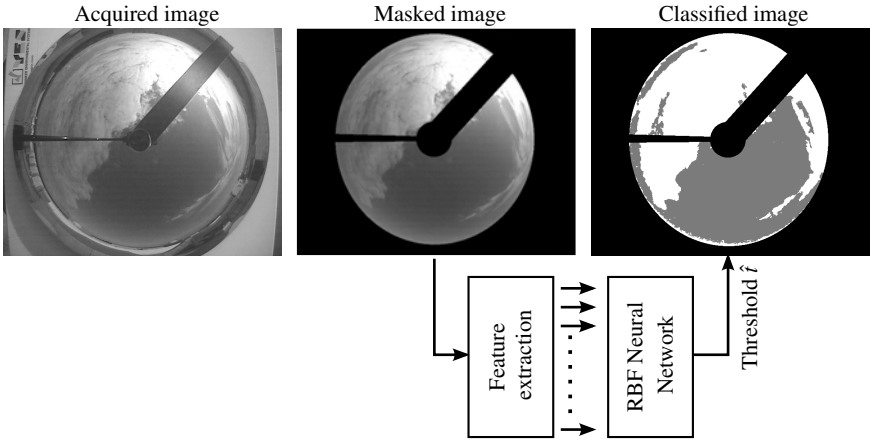

Fig. 13 Neural network image segmentation approach.

two additional RGB images; finally, from each RGB mode image, a grey intensity image was generated. This results in a total of 7 different images and 17 distinct intensity channels. From the latter, the sample mean, standard deviation, and skewness were extracted. Additionally, from the red and grey intensity channels (6 in total) histogram, the most frequent, first non-zero, and last non-zero intensity levels were also extracted.

From the lookup table, **F**, of 69 features, the model chromosomes were allowed to have $d \in [2,36]$ input terms. The number of neurons, n, was restricted to the interval $[2,24]$. As in the electricity consumption prediction problem, the model parameters were estimated via the LM algorithm using the modified training criterion (13) as outlined in Sect. 2.4. The initial centre locations for the Gaussian activation functions were selected randomly from the input patterns in $\mathbf{D}^t$, the corresponding initial spreads, σ_i, were determined by the rule (18), and the linear parameters were initialised using (12). The early stopping method was employed to stop the training algorithm by setting $k^{max} = 50$ in (20).

The MOGA was also employed to evolve a set of models whose selected number of neurons and input terms optimise a number of pre-specified goals and objectives. To this respect two objectives were set-up for minimisation: the RMS of the error computed on $\mathbf{D}^t$ and on $\mathbf{D}^g$, respectively denoted by $\rho(\mathbf{D}^t)$ and $\rho(\mathbf{D}^g)$. As the ANNs are randomly initialised, for each of them 25 training trials were executed and the average of both objectives was used for evaluation purposes. Recall that this procedure decreases the chance of one potentially good model being poorly evaluated due to a bad choice of initial parameters. Once the MOGA execution was terminated, for each of the preferable ANN models a larger number of training trials was executed in order to select one model for application. This choice was made by taking into account the actual objective values attained on each of the trials and also the RMS output error obtained on the validation data set.

The model that was selected from the identification experiment was compared to other thresholding methodologies [12], namely, a fix threshold approach, the Ridler, Calvard and Trussel (RCT) algorithm [35, 46], and Otsu's method [33]. For the first, an histogram based analysis was made in order to identify the best single threshold value that could be applied to all the 410 images. Global (over all the images) pixel intensity probability mass functions were separately computed for each of the classes, **S** and **C**. Then, a search was conducted in a vicinity around the intersection point of the PMFs in order to find the threshold minimising the average (over all images) value of (27). This was found to be $t = 158$ on the *hsvR* pixel intensity scale (the same used for the ANN approach).

The RCT method is an histogram-iteration form [46] of an iterative thresholding algorithm [35] that we denote by RCT in the following. A brief description of the method may be found in [12]. For a more in-depth view the reader should consult [35, 46, 30, 44]. Briefly, the RCT method tries to iteratively estimate the average pixel intensity of both classes and computes the threshold as the average of the classes sample mean.

The principle behind the method proposed by [33] is very simple: an exhaustive search is conducted on the pixel intensity scale for the threshold that maximises the inter-class variance. Again a brief overview may be found in [12], whereas for more detailed descriptions [33] or [44] may be consulted.

3.2.2 Results and Discussion The MOGA execution was stopped after 50 generations yielding 11 ANNs in the Pareto front as highlighted in the top-left plot of Fig. 14, where a detail of the objective values is shown. Regarding the number of neurons of the 11 selected models, four of them had from 12 to 14, the remaining seven had 22 or 23 neurons. Concerning the number of input features, the models employed from 29 to 36.

As mentioned before, 50 additional training trials were executed for each model selected. The resulting objective values are depicted in the top-right plot of Fig. 14. The plots at the bottom of the figure show the corresponding results considering the evaluation of each model structure instance on the validation data set: the RMS error obtained on $\mathbf{D}^t$ and $\mathbf{D}^g$ is plotted against the RMS error obtained in the validation data set, $\mathbf{D}^v$. The results marked with a dark square were obtained by the RBF ANN that was selected after analysis of all the results. It presented the most favourable balance in the objectives, achieving the RMS error values of 13.10, 13.12, and 14.65, respectively on $\mathbf{D}^t$, $\mathbf{D}^g$, and $\mathbf{D}^v$. It is a network with 30 inputs and 22 neurons.

Regarding the cloud cover fraction estimation, Table 3 presents the minimum, average, and maximum ε results obtained by the ANN model selected and by the remaining methods employed for comparison. For the ANN, they are presented considering the training and testing data sets together (involved in the MOGA ANN optimisation), the validation data set alone, and the three data sets altogether. It may be seen that the RCT and Otsu methods achieve similar results to those obtained with a fixed threshold. The results obtained by the RBF ANN selected using the framework presented in Sect. 2 represent an improvement in average accuracy of

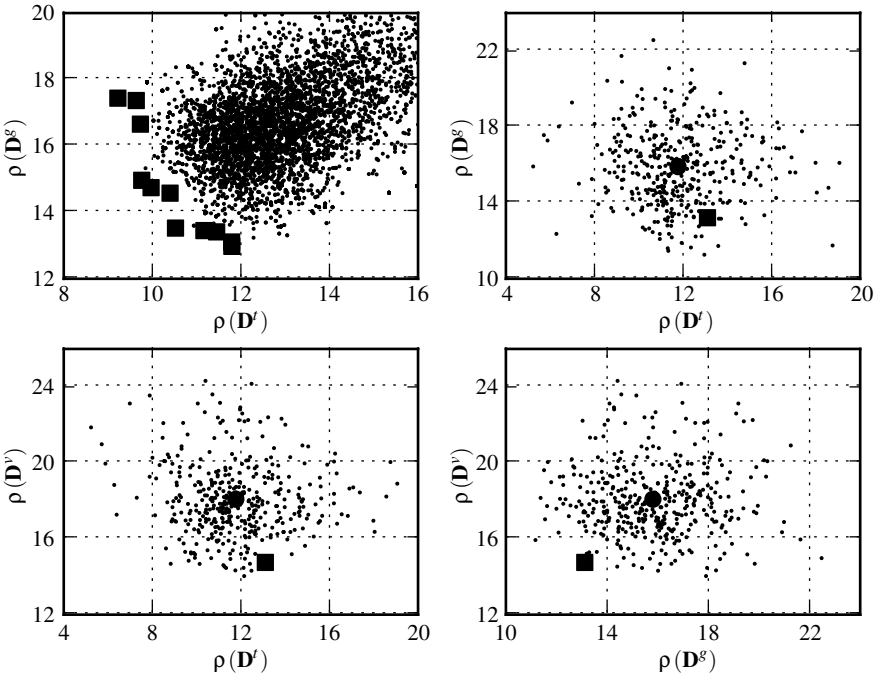

Fig. 14 MOGA results

Table 3 Absolute error of cloud cover estimation obtained by the RFB ANN image threshold approach (first three lines). Results obtained by three additional methods are shown for comparison (last three lines).

		minimum	average	maximum
Data set:	Training and testing	0.00	5.31	58.46
	Validation	0.00	4.74	43.71
	Altogether	0.00	5.22	58.46
Other methods:	Fixed threshold	0.00	11.24	82.64
	RCT method	0.00	11.34	98.21
	Otsu's method	0.00	11.07	63.59

approximately 50% when compared to the best results obtained by the remaining methods.

Figure 15 presents the absolute error values for the reference cloudiness of each image, where, for the ANN plot (bottom), the circles correspond to images in the training or testing data sets, and the dark squares to images in the validation data set. The similarity of results achieved by the three methods used for comparison is

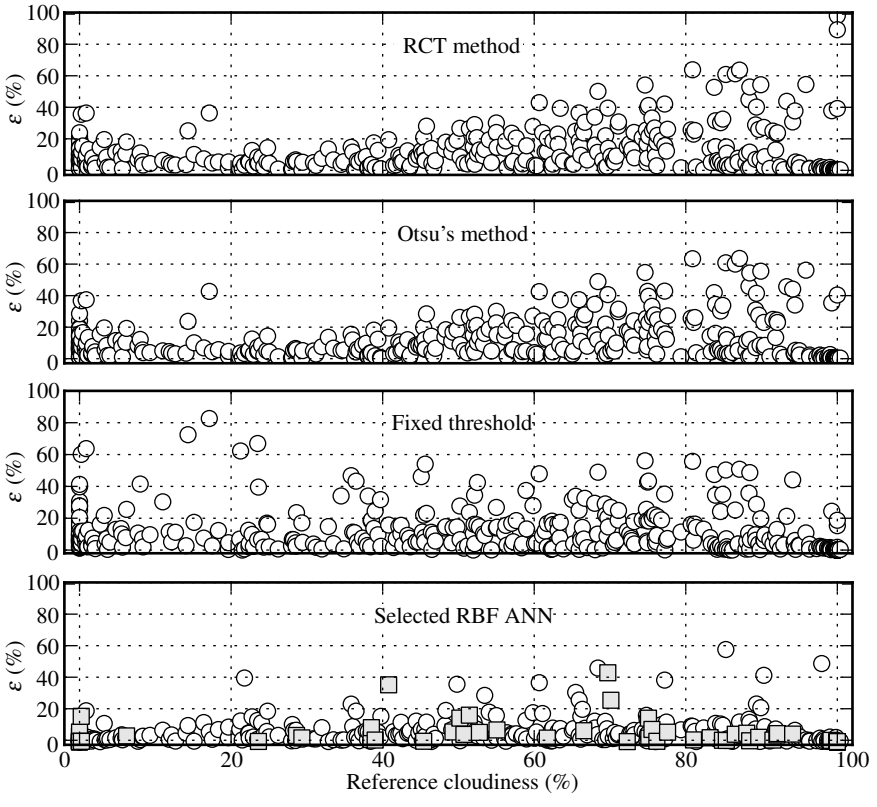

Fig. 15 Error performance of the three methods used for comparison, and of the RBF ANN image thresholding approach (bottom)

visible, although the fixed threshold approach exhibits improved uniformity of the error for the reference cloudiness when compared to the RCT and Otsu's methods. The improvement achieved by the selected ANN model is noticeable with most error values under 20%.

Despite the improvement obtained by the RBF ANN thresholding methodology there are a few directions in future work expected to further improve the results. Perhaps the most important, regarding the use of the MOEA to select ANNs, consists in specifying the objective space in a different way. In most images ε is not symmetric around the optimum threshold, thus minimising the threshold estimation error may not guarantee the best results. A better approach would consist in building a matrix where for each image (lines) the value of ε is computed for each pixel

intensity (columns), so that it becomes possible to map the NN threshold estimation to a cloud cover estimation error. The latter should be minimised in the MOGA search for NN structures. This is currently being implemented and will result in another iteration of the model design cycle. Once this is carried out, the resulting RBF ANN will be used to build a time-series of cloudiness from an existing time-series of GBAS images. Then a cloudiness predictive model will be identified and employed for the benefit of global solar radiation predictive models identification having cloudiness as an exogenous input. The goal is to conclude if that approach is preferable to auto-regressive solar radiation predictive models.

4 Concluding Remarks

Neural network modelling is an iterative process, requiring, at the present stage, substantial skills and knowledge from the designer. It is our view that the methodology presented in this article, employing multiobjective evolutionary algorithms for the design of neural models, is a suitable tool to aid the designer in this task. It incorporates inputs, model order and structure selection, as well as parameter estimation, providing the designer with a good number of well performing models with varying degrees of complexity. Importantly, the model identification framework is suitable, with minor adaptation, to most feed-forward artificial neural network methodologies. It also allows the incorporation of objectives which are specifically designed by considering the final application of the model. Through the analysis of the results obtained in one iteration, the search space can be reduced for future iterations, therefore allowing a more refined search in promising model regions. This was demonstrated in practice, by the presentation of two model identification experiments that were designed by taking into account results from previously executed experiments. In both, significant improvements were achieved not only when compared to previous work, but also by comparison with different methodologies.

Acknowledgements. The authors thank the Portuguese National Science Foundation for funding this work with project FCT PTDC/ENR/73345/2006, the Portuguese Power Grid company, "REN - Rede Eléctrica Nacional", for the funding and support, M. Eng. Rui Pestana from REN for the advice and collaboration, and Doctor Carlos M. Fonseca for the multiobjective genetic algorithm. The first author thanks the European Commission for the funding through grant PERG-GA-2008-239451.

References

1. Amari, S., Murata, N., Müller, K.R., Finke, M., Yang, H.: Asymptotic statistical theory of overtraining and cross-validation. IEEE Transactions on Neural Networks 8(5), 985–996 (1997), doi:10.1109/72.623200
2. Bauer, M., Buchtala, O., Horeis, T., Kern, R., Sick, B., Wagner, R.: Technical data mining with evolutionary radial basis function classifiers. Applied Soft Computing 9, 765–774 (2009), doi:10.1016/j.asoc.2008.07.007

3. Billings, S.A., Zheng, G.L.: Radial basis function network configuration using genetic algorithms. Neural Networks 8(6), 877–890 (1995)
4. Branke, J.: Evolutionary algorithms for neural network design and training. Tech. Rep. 322, University of Karlsruhe, Institute AIFB, Karlsruhe, Germany (1995)
5. Carse, B., Pipe, A.G., Fogarty, T.C., Hill, T.: Evolving radial basis function neural networks using a genetic algorithm. In: IEEE International Conference on Evolutionary Computation, vol. 1, pp. 300–305 (1995), doi:10.1109/ICEC.1995.489163
6. Chen, S., Wu, Y., Luk, B.: Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks. IEEE Transactions on Neural Networks 10(5), 1239–1243 (1999)
7. Coello Coello, C.: Recent trends in evolutionary multiobjective optimization. In: Jain, L., Wu, X., Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization, Advanced Information and Knowledge Processing, pp. 7–32. Springer, Heidelberg (2005), http://dx.doi.org/10.1007/1-84628-137-7_2, doi:10.1007/1-84628-137-7-2
8. Coello Coello, C.: Evolutionary multi-objective optimization: a historical view of the field. IEEE Computational Intelligence Magazine 1(1), 28–36 (2006), http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1597059&tag=1, doi:10.1109/MCI.2006.1597059
9. Crispim, E.M., Ferreira, P.M., Ruano, A.E.: Prediction of the solar radiation evolution using computational intelligence techniques and cloudiness indices. International Journal of Innovative Computing, Information and Control 4(5), 1121–1133 (2008)
10. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons., Ltd, Chichester (2001)
11. Ferreira, P.M., Faria, E., Ruano, A.E.: Neural network models in greenhouse air temperature prediction. Neurocomputing 43(1-4), 51–75 (2002)
12. Ferreira, P.M., Martins, I.A., Ruano, A.E.: Cloud and clear sky pixel classification in ground-based all-sky hemispherical digital images. In: Ferreira, P.M. (ed.) Proceedings of CMTEE 2010, the IFAC Conference on Control Methodologies and Technology for Energy Efficiency. International Federation of Automatic Control, Vilamoura, Portugal (2010)
13. Ferreira, P.M., Ruano, A.E.: Exploiting the separability of linear and non-linear parameters in radial basis function neural networks. In: IEEE Symposium 2000: Adaptive Systems for Signal Processing, Communications, and Control, Canada, pp. 321–326 (2000), http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=882493, doi:10.1109/ASSPCC.2000.882493
14. Ferreira, P.M., Ruano, A.E.: Application of computational intelligence methods to greenhouse environmental modelling. In: (WCCI 2008) IJCNN 2008 (IEEE World Congress on Computational Intelligence). 2008 IEEE International Joint Conference on Neural Networks, pp. 3582–3589 (2008), doi:10.1109/IJCNN.2008.4634310
15. Ferreira, P.M., Ruano, A.E.: On-line sliding-window methods for process model adaptation. IEEE Transactions on Instrumentation and Measurement 58(9), 3012–3020 (2009), doi:10.1109/tim.2009.2016818
16. Ferreira, P.M., Ruano, A.E., Fonseca, C.: Genetic assisted selection of rbf model structures for greenhouse inside air temperature prediction. In: IEEE Conference on Control Applications, Turkey, pp. 576–581 (2003)
17. Ferreira, P.M., Ruano, A.E., Fonseca, C.: Evolutionary multi-objective design of radial basis function networks for greenhouse environmental control. In: IFAC World Congress on Automatic Control 16th, Czech Republic (2005)

18. Ferreira, P.M., Ruano, A.E., Pestana, R.: Improving the identification of rbf predictive models to forecast the portuguese electricity consumption. In: Ferreira, P.M. (ed.) Proceedings of CMTEE 2010, the IFAC Conference on Control Methodologies and Technology for Energy Efficiency. International Federation of Automatic Control, Vilamoura, Portugal (2010)
19. Ferreira, P.M., Ruano, A.E., Pestana, R., Kóczy, L.T.: Evolving rbf predictive models to forecast the portuguese electricity consumption. In: ICONS 2009: The 2nd IFAC Int. Conference on Intelligent Control Systems and Signal Processing, Istanbul, Turkey (2009)
20. Fletcher, R.: Practical Methods of Optimization, 2nd edn. Wiley Interscience, Hoboken (2000)
21. Fonseca, C., Fleming, P.: Non-linear system identification with multiobjective genetic algorithms. In: Proceedings of the 13 IFAC World Congress, vol. C, pp. 187–192 (1996)
22. Fonseca, C., Fleming, P.: Multiobjective optimization and multiple constraint handling with evolutionary algorithms i: A unified formulation. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans 28(1), 26–37 (1998), doi:10.1109/3468.650319
23. Gill, P.E., Murray, W., Wright, M.H.: Practical Optimization. Academic Press, Inc., London (1981)
24. Griffin, I., Fleming, P.: An overview of non-linear identification and control with neural networks. In: Ruano, A.E. (ed.) Intelligent Control Using Soft-Computing Methodologies. Control Series, pp. 89–118. IEE Publishing (2005)
25. Guillén, A., Pomares, H., González, J., Rojas, I., Valenzuela, O., Prieto, B.: Parallel multiobjective memetic rbfnns design and feature selection for function approximation problems. Neurocomputing 72(16-18), 3541–3555 (2009), doi:10.1016/j.neucom.2008.12.037
26. Haykin, S.: Neural Networks: a Comprehensive Foundation, 2nd edn. Prentice Hall, Inc., Englewood Cliffs (1999)
27. Jung, J., Reggia, J.: Evolutionary design of neural network architectures using a descriptive encoding language. IEEE Transactions on Evolutionary Computation 10(6), 676–688 (2006), doi:10.1109/TEVC.2006.872346
28. Kaylani, A., Georgiopoulos, M., Mollaghasemi, M., Anagnostopoulos, G.C., Sentelle, C., Zhong, M.: An adaptive multiobjective approach to evolving art architectures. IEEE Transactions on Neural Networks 21(4), 529–550 (2010)
29. Lee, C.W., Shin, Y.C.: Growing radial basis function networks using genetic algorithm and orthogonalization. International Journal of Innovative Computing, Information and Control 5(11(A)), 3933–3948 (2009)
30. Leung, C., Lam, F.: Performance analysis for a class of iterative image thresholding algorithms. Pattern Recognition 29(9), 1523–1530 (1996)
31. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial and Applied Mathematics 11(2), 431–441 (1963)
32. McDonnell, J., Waagen, D.: Determining neural network hidden layer size using evolutionary programming. In: Proceedings of the 1993 World Congress on Neural Networks, vol. III, pp. 564–657 (1993)
33. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man and Cybernetics SMC-9(1), 62–66 (1979)
34. Peck, C., Dhawan, A., Meyer, C.: Genetic algorithm based input selection for a neural network function approximator with applications to ssme health monitoring. In: IEEE International Conference on Neural Networks, vol. 2, pp. 1115–1122 (1993), doi:10.1109/ICNN.1993.298714

35. Ridler, T., Calvard, S.: Picture thresholding using an iterative selection method. IEEE Transactions on Systems, Man and Cybernetics SMC-8(8), 630–632 (1978)
36. Rodríguez-Vázquez, K., Fonseca, C., Fleming, P.: Identifying the structure of nonlinear dynamic systems using multiobjective genetic programming. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans 34(4), 531–545 (2004), doi:10.1109/TSMCA.2004.826299
37. Ruano, A., Crispim, E., Frazão, P.: Moga design of neural network predictors of inside temperature in public buildings. In: Balas, V., Fodor, J., Várkonyi-Kóczy, A. (eds.) Soft Computing Based Modeling in Intelligent Systems. SCI, vol. 196, pp. 35–61. Springer, Heidelberg (2009), doi:10.1007/978-3-642-00448-3-3
38. Ruano, A., Fleming, P., Jones, D.: Connectionist approach to pid autotuning. IEE Proceedings (part D), 139(3), 279–285 (1992), http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=141517
39. Ruano, A., Jones, D., Fleming, P.: A new formulation of the learning problem of a neural network controller. In: Proceedings of the 30th IEEE Conference on Decision and Control, vol. 1, pp. 865–866 (1991), http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=261439, doi:10.1109/CDC.1991.261439
40. Ruano, A.E., Ferreira, P.M., Cabrita, C., Matos, S.: Training neural networks and neurofuzzy systems: A unified view. In: Proceedings of the 15th IFAC World Congress, vol. 15 (2002)
41. Ruano, A.E., Ferreira, P.M., Fonseca, C.: An overview of non-linear identification and control with neural networks. In: Ruano, A.E. (ed.) Intelligent Control Using Soft-Computing Methodologies. Control Series, pp. 37–87. IEE Publishing (2005)
42. Ruano, A.E., Ferreira, P.M., Mendes, H.: Moga design of temperature and relative humidity models for predictive thermal comfort. In: Ferreira, P.M. (ed.) Proceedings of CMTEE 2010, the IFAC Conference on Control Methodologies and Technology for Energy Efficiency. International Federation of Automatic Control, Vilamoura, Portugal (2010)
43. Ruano, A.E., Fleming, P., Teixeira, C., Rodríguez-Vázquez, K., Fonseca, C.: Nonlinear identification of aircraft gas-turbine dynamics. Neurocomputing 55(3-4), 551–579 (2003), doi:10.1016/S0925-2312(03)00393-X
44. Sezgin, M., Sankur, B.: Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging 13(1), 146–165 (2004)
45. Sjöberg, J., Ljung, L.: Overtraining, regularization, and searching for minimum with application to neural networks. In: Preprint IFAC Symposium on Adaptive Systems in Control and Signal Processing, pp. 669–674 (1994)
46. Trussel, H.: Comments on picture thresholding using an iterative selection method. IEEE Transactions on Systems, Man and Cybernetics SMC-9(5), 311 (1979)
47. Yen, G.: Multi-objective evolutionary algorithm for radial basis function neural network design. In: Jin, Y. (ed.) Multi-objective machine learning. SCI, vol. 16, pp. 221–239. Springer, Heidelberg (2006), doi:10.1007/3-540-33019-4-10
48. Zitzler, E., Laumanns, M., Bleuler, S.: A tutorial on evolutionary multiobjective optimization. In: Gandibleux, X., et al. (eds.) Metaheuristics for Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems, vol. 535, pp. 3–37. Springer, Heidelberg (2004)