Neural Comput & Applic
DOI 10.1007/s00521-011-0582-x

ORIGINAL ARTICLE

Evolutionary product unit neural networks for short-term wind speed forecasting in wind farms

C. Hervás-Martínez · S. Salcedo-Sanz · P. A. Gutiérrez · E. G. Ortiz-García · L. Prieto

Received: 6 July 2010 / Accepted: 17 March 2011

Abstract Combinations of physical and statistical wind speed forecasting models are frequently used in wind speed prediction problems arising in wind farms management. Artificial neural networks can be used in these models as a final step to obtain accurate wind speed predictions. The aim of this work is to determine the potential of evolutionary product unit neural networks (EPUNNs) for improving the accuracy and interpretation of these systems. Traditional neural network and EPUNN approaches have been used to develop different wind speed prediction models. The results obtained using different EPUNN models show that the functional model and the hybrid algorithms proposed provide very accurate prediction compared with standard neural networks used to solve this regression problem. One of the main advantages of the application of these EPUNNs has been the possibility of obtaining some interpretation of the non-linear relation predicted by the model, as will be shown in real data of a wind farm in Spain.

Keywords Short-term wind speed forecasting · Product unit neural networks · Evolutionary programming

C. Hervás-Martínez · P. A. Gutiérrez
Department of Computer Science and Numerical Analysis, Universidad de Córdoba, Córdoba, Spain

S. Salcedo-Sanz (✉) · E. G. Ortiz-García
Department of Signal Theory and Communications, Universidad de Alcalá, Alcalá de Henares, Spain
e-mail: sancho.salcedo@uah.es

L. Prieto
Energy Resources Department, Iberdrola Renovables, Madrid, Spain

1 Introduction

Renewable energy in Europe is considered as a priority line of actuation, with a target to achieve 20% of the energy consumption from renewable sources by 2020. Specifically, it is expected that over 16% of the electricity generated in Europe comes from wind energy plants by that date. Also, this value situates the wind power as one of the main alternative sources of energy, intensively supporting the research in technologies involved in wind farms [1–5]. One of the main problems of wind energy is the integration of this type of energy in the electric transport networks, which must be improved to reach the proposed objectives. Among these problems is remarkable the management of the energy production, influenced by the variability of the wind in different months or seasons. In this scenario with a high integration of wind energy in the transport network, the prediction of the generated power in wind farms is a key problem, both for the producers and managers of the transport network. The different agents that manage the national energy transportation networks must have a knowledge of the production variability as soon as possible, to be able to plan the rest of energy supplies, maintaining the integrity of the whole system. From the producers point of view, the prediction of the wind energy is not only a compulsory procedure, but also the profitability of their projects is affected by their ability to predict the generated power, because in many countries they are penalized for failing their predictions. In addition, in a future with a greater integration of wind power, it is expected an increasing in the penalties for wrong predictions, and thus the problem of wind power prediction in wind farms will be crucial.

Nowadays, modern forecasting models for power prediction in wind farms are based on the combination of

Published online: 01 April 2011

physic and statistical models. The physic models can be global, meso-scale, or even local models, taking into account the specific local orography of the wind farm. Statistical models are usually included in the prediction systems instead of local physic models, and it has been shown that they produce a significant improvement in the prediction over physical approaches. Generally speaking, there exist two possibilities to make short-term predictions in wind farms: the first one is to make a direct prediction of the produced power in the farm considering all the wind turbines at the same time and different conditions in the farm (meteorology, etc.). Another way to tackle this problem (more interesting from the point of view of management) is to carry out a wind prediction in each wind turbine of the farm and to convert it into expected power production by means of the power curve of the turbine. This last way of prediction provides a better management of the wind farms, because it allows to evaluate the production under a priori known conditions, such as inoperative or intentionally stopped wind turbines to prevent possible damages.

On the other hand, wind speed forecasts in each wind turbine can be tackled from two different points of view: wind speed forecasts based on past wind data and wind speed forecasts using meteorological models to include the environment dynamics. Although there are several studies based on past data [6, 7] (note that the acquisition of past data is simple and the application of any regression technique is straightforward), this kind of prediction is usually applied only for long-term forecasting problems (monthly, quarterly, yearly), and it hardly ever provides good results in short-term prediction. For short-term prediction, the use of meteorological models (which include information of atmospheric dynamics) is essential to obtain competitive results [8, 9]. In these previous works, different regression approaches based on artificial intelligence or other traditional methods are used, such as artificial neural networks [3, 10, 11], support vector machines [12], ARMA models [13], and Kalman filters [4].

In this paper, the potential of the product unit neural networks in short-term wind speed forecasting in several wind turbines using meteorological models is studied. These networks are optimized using different memetic evolutionary algorithms. In this way, a methodology for obtaining the structure and weights of a product unit neural network is proposed, based on the combination of an evolutionary programming algorithm, a clustering process and a local improvement procedure carried out by the Levenberg–Marquardt algorithm. The multiplicative neural networks are among the most recent and interesting neural networks models, and they contain nodes that multiply their inputs instead of adding them, which allows inputs to interact non-linearly. This class of multiplicative neural networks comprises such types as sigma-pi networks and product unit (PU) networks [14]. This paper use three-layer networks, where only the hidden layer consists of PUs, while the output layer has additive units. Both layers use linear activation functions. Advantages of product unit neural networks (PUNNs) are increased information capacity and the ability to form higher-order combinations of the inputs. Also, it is possible to obtain upper bounds of the Vapnik-Chervonenkis dimension of PUNNs similar to those obtained for sigmoidal neural networks [15]. Another interesting property of PUNNs is that they can approximate any function with a given accuracy as well as sigmoidal neural networks [16]. Product unit based networks have a major drawback: their training is more difficult than the training of standard sigmoidal based networks [14]. The main reason for this difficulty is that networks based on product units have more local minima and more probability of becoming trapped in them [17]. The flexibility of PUNNs when compared with sigmoidal additive NNs encourages us to select this type of network for the wind speed prediction problem, avoiding the problems mentioned by using an advanced evolutionary algorithm.

The rest of the paper has the following structure: in the next section we present in detail the hybrid system proposed, describing the global model used, the first down-scaling carried out (physical downscaling) with the MM5 model and the artificial neural network used to perform the final statistical downscaling. The potential of evolutionary PUNNs for this last step is evaluated in this paper, their learning and structure being described in Sect. 3. Section 4 presents the experiments carried out to evaluate the performance of our approach. The short-term (48 h horizon) wind speed forecast in a wind park with 33 wind turbines, located at the southeast of Spain, is considered. Section 5 gives some final remarks for concluding the paper.

2 Hybrid model considered for short-term wind speed prediction

A system of short-term wind speed forecast based on global and mesoscale models has been recently presented in [8] (Fig. 1). The original system starts from a given global weather forecast model, and two different processes of down-scaling are considered, the first one a physical down-scaling using a mesoscale forecasting model, and the second one a statistical downscaling processing, using a multi-layer perceptron. This last process of down-scaling can be, of course, replaced by any other type of regressor. In this paper, we study the modification of this hybrid system for wind speed prediction to include PUNNs, the main characteristics of these networks and how they can improve the performance of the system.

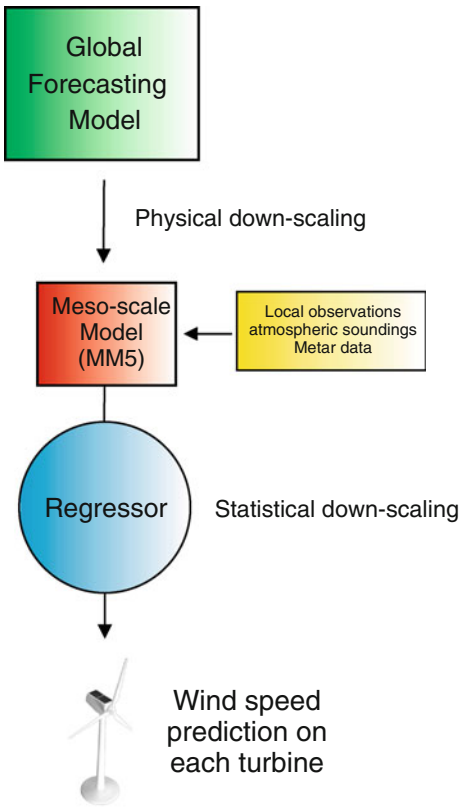

Fig. 1 Outline of the general form of a hybrid system for short-term wind speed forecasting

The system proposed in [8] uses a global prediction system (specifically the Global Forecasting System, GFS [18]) to obtain a first prediction of the meteorological variable for future times at given positions and altitudes, considering as horizontal domain the entire Earth. The global prediction system integrates the Navier–Stokes equations, to provide a set of atmospheric variables which can be useful for different applications, such as pressure (P), temperature (T), geopotential height (gph) and also wind speed and direction ($\mathbf{v}$). In general, these variables are solved for a given number of levels in height, which usually vary between 10 and 1,000 hPa. In addition, the majority of the global models also provide these basic variables at a level of 10 m over the ground. Regarding the spatial resolution of the model proposed in [8] was $1° \times 1°$. Note that the forecasts from global models do not include local characteristics, so a down-scaling process is carried out.

Two processes of down-scaling are considered in [8]. First, a process of down-scaling (physical down-scaling) is performed, using the Fifth generation Mesoscale Model (MM5 model) [19]. The MM5 is a *limited area* model, which solves the Navier-Stokes equations which modelled the behaviour of the atmosphere (similar to the global models), but without including ocean-land interactions and other important variables of the global forecasting models.

The MM5 initialization is carried out by using the data from the global prediction systems considered, and also data from atmospheric soundings, using aerostatic balloons at the Iberian Peninsula in the following sites: Gibraltar, Madrid, Murcia, Palma de Mallorca, Santander, and Zaragoza. In addition, also *metar* data are included in the MM5 initialization. Metar are surface data measured in the 39 airports of the Iberian Peninsula and the Balear Islands, each 30 min. Variables included in metar data are pressure, temperature, wind speed and direction, among others. The result of this physical down-scaling carried out using the MM5 model is a forecast of the wind speed and direction in a more realistic orography than the one given by the global forecasting models. The MM5 model can be operated using different parametrizations, and this provides a number of possible different models to feed the second part of the down-scaling process, the statistical down-scaling using neural networks.

This final step consists in applying a regressor (a neural network in our case) to the data from the MM5 models, in order to obtain the wind speed prediction in each turbine of the wind farm. In this paper, as it has been mentioned before, we substitute the MLP in [8] by a PUNN, which is trained by a dynamic hybrid evolutionary programming algorithm with clustering (DHEPC) [20]. The PUNN is trained to obtain the best quality wind speed prediction. The DHEPC algorithm is based on the combination of an evolutionary algorithm, a clustering process and a gradient descent local search procedure. The three elements are used for architectural design and estimation of weights. This synergy among diverse optimization methods can provide different families of hybrid algorithms where the search is based on a first step of exploration, followed by a second step of exploitation. The benefits of mutual interactions between a local and a global optimization method working together have been studied in computational science, giving rise to techniques which have been called memetic algorithms (MAs), hybrid evolutionary algorithms HEAs, Lamarckian EAs, Balwinian EAs, etc. [22–24].

3 Evolutionary product unit neural networks

In this paper, different hybrid evolutionary programming (HEP) methodologies are considered for obtaining the structure and weights of the proposed neural networks based on product units. The methodologies are based on the combination of an EP algorithm (global explorer), a clustering process and a local improvement procedure carried out by the Levenberg–Marquardt (L–M) algorithm (local exploiter). The aim of the proposed approaches is the automatic optimization of the structure and weights of PUNNs used for the determination of wind speed forecasts.

Note that the PUNN models are feed with information extracted from the MM5 models.

Different versions of an HEP algorithm are used depending on how the local search and the cluster partitioning are combined with the EP algorithm:

- The first one consists of the application of an *Evolutionary Programming* algorithm (EP) without the clustering process nor the local search. The EP paradigm is selected, which discards the use of recombination, since crossover is usually regarded as less effective for network evolution [25].
- We also consider the application of an HEP approach, where the EP is run and the local search is only applied to the final best solution obtained in the evolutionary process (*Hybrid Evolutionary Programming optimizing the Final best solution*, HEPF). This allows the precise local optimum around the final solution to be found. The local search is not applied to all the individuals of the population due to the high computational cost that it represents and because it could result in over-fitting training data.
- The last alternative is the so-called *Dynamic Hybrid Evolutionary Programming with Clustering* (DHEPC), which carries out a clustering process in a specific space where each individual is mapped to a different point depending on its performance. After that, the L–M algorithm is applied to the best individual in each cluster dynamically every G_t generations, where G_t must be fixed by the user. The optimized individuals are stored and the final solution is the best one among the local optima found during the evolutionary process. This combination of a clustering process and a local optimization method for EAs has been previously proposed, reporting a good performance [20]. The process considers a large enough subset of the best individuals of the population and it is very important to determine which rate of the best individuals is going to be considered and the value for the number of clusters. Thus, the local optimization procedure is carried out in several generations of the EA. The individuals do not return to the population, because crossover operator is not used (and the only way the information of the optimized individuals will contribute anything for future generations is by using a crossover operator). Instead, optimized individual will be stored in a pool of individuals, and the best final one will be returned as solution.

3.1 Product unit neural networks

In this paper, we consider a procedure to construct regressors based on learning Product Unit Neural Networks (PUNNs) from the patterns of the training set. This kind of models is an alternative to Multilayer Perceptrons (MLPs), considering multiplicative hidden nodes instead of additive ones. In this way, the output of each hidden node ($B_j(\mathbf{x}, \mathbf{w}_j)$) is the following:

$$B_j(\mathbf{x}, \mathbf{w}_j) = \prod_{i=1}^{m} x_i^{w_{ji}} \tag{1}$$

where $\mathbf{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jm})$ is the set of exponents for the jth hidden node. The proposed model is a linear combination of p of these basis functions:

$$y = f(\mathbf{x}, \boldsymbol{\theta}) = \beta_0 + \sum_{j=1}^{p} \beta_j \left(\prod_{i=1}^{m} x_i^{w_{ji}} \right) \tag{2}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \mathbf{w}_1, \ldots, \mathbf{w}_p\}$ is the vector of parameters of the model, $\mathbf{x} \in S \subset \mathbb{R}^m$ and S is a sub-set of the real space of dimension m with $x > 0, \forall x \in \mathbf{x}$. The vector $\boldsymbol{\beta} = \{\beta_0, \beta_1, \ldots, \beta_p\}$ stands for the set of coefficients multiplying each of the basis functions, which are represented by the vector $\mathbf{B}_i = \{B_0(\mathbf{x}_i, \mathbf{w}_0), B_1(\mathbf{x}_i, \mathbf{w}_1), \ldots, B_p(\mathbf{x}_i, \mathbf{w}_p)\}$, where $B_0(\mathbf{x}, \mathbf{w}_0) = 1$ is the bias of the estimated output, and $\mathbf{w}_j$ is the vector of the coefficients associated with the jth basis function.

For the data set $D = \{\mathbf{x}_l, y_l\}$, for $l = 1, \ldots, n$, the regression model can be expressed by means of a potential base function topology, such that

$$y_l = f(\mathbf{x}_l, \boldsymbol{\theta}) = \beta_0 + \sum_{j=1}^{p} \beta_j \left(\prod_{i=1}^{m} x_i^{w_{ji}} \right) + e_l, \quad l = 1, \ldots, n, \tag{3}$$

where $\beta_j, w_{ji} \in \mathbb{R}$ and e_l is the error obtained for the lth pattern.

If the usual methods of estimation of parameters of a linear model are followed, in this case the procedure is not trivial because the design matrix $\mathbf{B} = \{\mathbf{B}_1, \ldots, \mathbf{B}_n\}$ depends on the w_j parameters and the elements of the $\mathbf{B}$ matrix are potential functions of the $\mathbf{x}$ values. Consequently, a hybrid EA is used to estimate the optimum values of p and those of the β_j and w_j coefficients. The expression of (2) can be represented by a neural network with a proper architecture. For the wind speed prediction problem, the architecture is the following: one input layer consisting of six intput neurons (corresponding to the data of the MM5 model); one hidden layer with an appropriate number of nodes, which will be adjusted by the evolutionary algorithm; and one output layer with one output neuron providing the wind speed for each turbine. An example of this architecture can be seen in Fig. 2.

3.2 Evolutionary training

In this section, different versions of HEP algorithms are presented. First of all, the base EP algorithm is described.

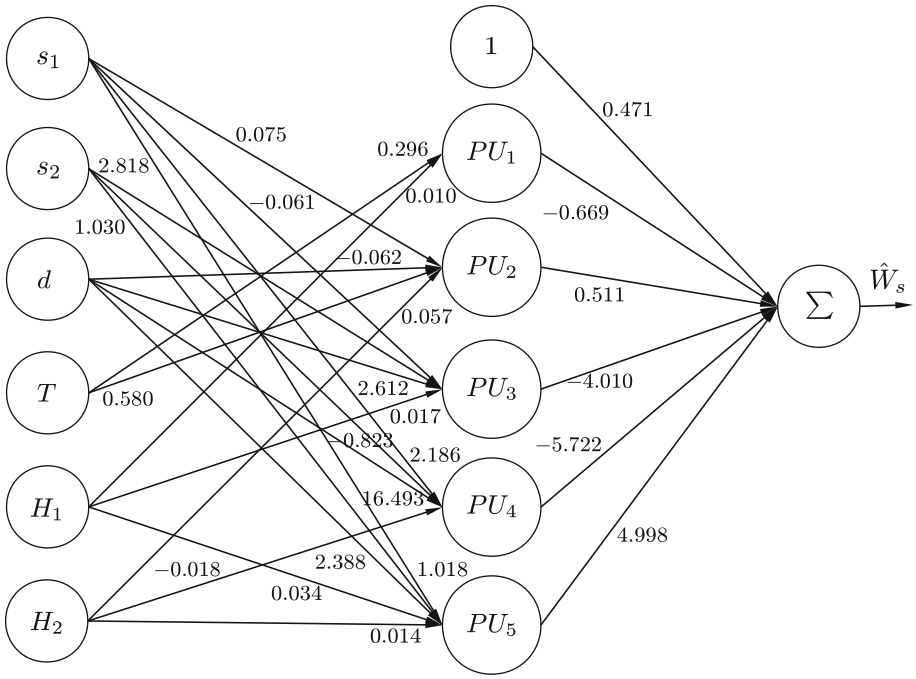

Fig. 2 Best model for Turbine #1

Fig. 3 Pseudo-code for the evolutionary programming (*EP*) algorithm

1. Generate initial population B of solutions, where each solution is a PUNN with random structure and weights
2. $g \leftarrow 0$
3. While $g < 400$ do:
   (a) $b \leftarrow \{the\ best\ of\ B\}$
   (b) Select $B' \subset B$, where B' is the best 10% of individuals from B.
   (c) Select $B'' \subset B$, where B'' is the best 90% of individuals from B.
   (d) Apply structural mutation to every individual of B' to produce B'_{struc}
   (e) Apply parametric mutation to every individual of B'' to produce B''_{param}
   (f) $B''' = B'_{struc} \bigcup B''_{param}$
   (g) $B = B'' \bigcup b$
   (h) $g \leftarrow g + 1$
   (i) **if** for 10 generations there was no improvement either in the average performance of the best 20% of the population or in the fitness of the best individual **then** $g \leftarrow 400$
4. End while
5. Return the best individual obtained.

Then, we describe how the local search and the cluster partitioning are combined with the EP algorithm.

3.2.1 Base algorithm: evolutionary programming (EP)

The population-based evolutionary algorithm used for PUNN architecture design and estimation of the real-parameters has common points with other evolutionary algorithms in the bibliography [26, 27]. It begins the search with an initial population of PUNNs, and, in each iteration, the population is updated. The population is subject to the operations of replication and mutation. With these features, the algorithm falls into the class of evolutionary programming (EP). A general pseudo-code for the proposed algorithm is shown in Fig. 3.

The high-level template of an evolutionary algorithm is a generation with three main components: selection, recombination, and replacement. In the proposed algorithms, the recombination is discarded in favour of two types of mutation operators: parametric and structural mutation. Crossover is not used due to its potential disadvantages in evolving artificial networks, because of the traditional belief that they generally disrupt the distributed functionality of the evolving solutions. The notion that crossover will be especially disruptive when a genetic representation is used which has a many-to-one mapping between genotype and phenotype has become known as the "permutation problem" [25]. The mutation operators generate a new solution by partly modifying an existing solution; thus, in parametric mutation the values of the coefficients of the model are changed

adding a Gaussian random value to a gene of the chromosome chosen at random; whereas in the structural mutation, the new solution is obtained adding or removing addends in the model of network (nodes); and adding, removing and/or re-defining coefficients of the model (connections).

The algorithm evolves architectures and connection weights simultaneously, each individual being a fully specified PUNN. The neural networks are represented using an object-oriented approach and the algorithm deals directly with the PUNN phenotype. Each connection is specified by a binary value indicating if the connection exists, and a real value representing its weight. As the crossover is not considered, this object-oriented representation does not assume a fixed order between the different PUs.

The EP algorithm is based on the following operators:

- *Initialization of the population*. For the generation of each network, the number of nodes in the hidden layer is an integer value in the $[M_{\text{MIN}}, M_{\text{MAX}}]$ interval. The number of connections between each node of the hidden layer and the input nodes is determined from a uniform distribution in the interval $(1, k]$, where k is the number of independent variables. Once the topology of the network is defined, each connection is assigned a weight from a uniform distribution in the interval $[-L, L]$ for the weights between the input and hidden layers, and in the interval $[-M, M]$ for the weights between the hidden layer and the output node. The normalization range for input and output variables is $[0.1, 0.9]$. In this form, we obtain the initial population B of PUNNs (see Fig. 3).
- *Fitness measure*. Let $D = \{(\mathbf{x}_l, y_l),\ l = 1, 2, \ldots, n\}$ be the training data set. We consider the mean square error of an individual $f(\mathbf{x}, \boldsymbol{\theta})$ [defined using (2)] of the population:

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{l=1}^{n}(y_l - f(\mathbf{x}_l, \boldsymbol{\theta}))^2 \tag{4}$$

and we define the fitness function $A(\boldsymbol{\theta})$ as the following increasing transformation of the $J(\boldsymbol{\theta})$ error:

$$A(\boldsymbol{\theta}) = \frac{1}{1 + J(\boldsymbol{\theta})} \tag{5}$$

- *Parametric mutation*. Parametric mutation uses of a simulated annealing algorithm. The severity of a mutation to an individual $f(\mathbf{x}, \boldsymbol{\theta})$ is dictated by the temperature, $T(f(\mathbf{x}, \boldsymbol{\theta}))$, given by:

$$T(f(\mathbf{x}, \boldsymbol{\theta})) = 1 - A(f(\mathbf{x}_l, \boldsymbol{\theta})), \quad 0 \le T(f(\mathbf{x}, \boldsymbol{\theta})) < 1 \tag{6}$$

Parametric mutation is accomplished for each coefficient w_{ji}, β_j of the model in (2) by adding Gaussian noise, where the variance of the normal distribution depends on the temperature:

$$w_{ji}(t+1) = w_{ji}(t) + \xi_1(t); \quad \beta_j(t+1) = \beta_j(t) + \xi_2(t) \tag{7}$$

where $\xi_k(t) \in N(0, \alpha_k(t) \cdot T(f(\mathbf{x}, \boldsymbol{\theta}))), \quad k \in \{1, 2\}$, represents a normal random variable with mean 0 and standard deviation $\alpha_k(t) \cdot T(f(\mathbf{x}, \boldsymbol{\theta}))$. The parameters $\alpha_k(t), k \in \{1, 2\}$, allow the adaptation of the learning process, changing along the evolution. There are different methods to update these parameters. We use one of the simplest methods, the 1/5 success rule of Rechenberg:

$$\alpha_k(t+1) = \begin{cases} (1+\beta)\alpha_k(t), & \text{if } A(g) > A(g-1), \quad \forall g \in \{t, t-1, \ldots, t-\rho\} \\ (1-\beta)\alpha_k(t), & \text{if } A(g) = A(g-1), \quad \forall g \in \{t, t-1, \ldots, t-\rho\} \\ \alpha_k(t) & \text{otherwise} \end{cases} \tag{8}$$

where $k \in \{1, 2\}, A(g)$ is the fitness of the best individual in generation g, ρ is the number of generations that is going to be analyzed when considering each update and β is the ratio of increment or decrement for the variances. It should be pointed that the modification of the exponents w_{ji} is different from the modification of the coefficients β_j, $\alpha_1(t) \ll \alpha_2(t)$, $\forall t$.

- *Structural mutation*. Structural mutation implies a modification of the structure of the function and allows the explorations of different regions of the search space, helping to keep the diversity of the population. There are five different structural mutation: node addition, node deletion, connection addition, connection deletion and node fusion. For each mutation (except for node fusion) there is a minimum value, δ_{MIN}, and a maximum value, δ_{MAX}, and the number of elements (nodes and connections) involved in the mutation is calculated as:

$$\Delta_{\text{MIN}} + \lfloor uT(f(\mathbf{x}, \boldsymbol{\theta}))\ (\Delta_{\text{MAX}} - \Delta_{\text{MIN}}) \rfloor \tag{9}$$

where u is a random uniform variable in the interval $[0, 1]$. In the node fusion, two randomly selected nodes, a and b, are replaced by a node c, which is a combination of both. The connections that are common to both nodes are kept, with a weight given by:

$$\beta_c = \beta_a + \beta_b; \quad w_{ic} = \frac{w_{ia} + w_{ib}}{2} \tag{10}$$

The connections that are not shared by the nodes are inherited by c with probability of 0.5 and its weight is unchanged. All the above mutations are made sequentially in the given order, with probability $T(f(\mathbf{x}_l, \boldsymbol{\theta}))$, in the same generation on the same network. If the probability does not select any mutation, one of the

mutations is chosen at random and applied to the network. For more details see [28].

3.2.2 *Hybrid evolutionary programming optimizing the final best solution (HEPF)*

EAs are a class of optimization algorithms, based in a population of solutions, which are efficient for exploring the entire search space. They are, however, relatively poor at finding the precise local optimal solution in the region into which the algorithm converges. Many researchers have shown that EAs perform well for global searching because they are capable of quickly finding and exploiting promising regions of the search space, but they take a relatively long time to converge to a local optimum [29].

In the past decade, new approaches have been reported for improving EAs using local improvement procedures (LIPs), which search a "neighbourhood" of the starting solution until either the first improvement or the best improvement (local optimum) is found. These new methodologies are based on the combination of LIPs, which are good at finding local optima (local exploiter), and evolutionary algorithms (global explorer). In this paper, we hybridize the EP algorithm previously presented with a local search Quasi-Newtonian algorithm, the Levenberg–Marquardt (L–M) method, because it is especially suitable for regression or forecasting problems where the sum-of-squares residual should be minimized. However, the particular combination of EA with local search is extremely important in terms of possible solution quality and computational efficiency; and therefore, the right mixture of local exploitation versus global exploration should be found.

The first proposal (hybrid evolutionary programming optimizing the Final best solution, HEPF) is the simplest approach, i.e., the L–M algorithm is applied to the best individual obtained by the EP algorithm in the final generation. This allows to refine the solution obtained by the EP algorithm, improving the final accuracy of the model.

3.2.3 *Dynamic hybrid evolutionary programming with clustering (DHEPC)*

Martınez-Estudillo et al. [20] proposed the hybrid combination of three methods for the design of evolutionary PUNNs (EPUNNs) for regression: an EA, a clustering process and a local search procedure. Clustering methods create groups (clusters) of mutually close points that could correspond to relevant regions of attraction. The clustering process is applied in a specific space where each individual is mapped to a different point depending on its performance. Then, local search procedures can be started once in every such region, e.g., from its centroid or from the best individual of the cluster. The authors of the work reported that the application of a clustering algorithm for selecting individuals representing the different regions in the search space was more efficient than using the optimization algorithm for every individual in the population.

In this paper, we apply this method (which is known as dynamic hybrid evolutionary programming with Clustering, DHEPC) to the problem of wind speed prediction. The clustering process is applied to the best $\tilde{s} \cdot N_{\mathrm{P}}$ individuals of the population, where $\tilde{s}$ is a parameter of the algorithm. The population is divided into K clusters $C_1, C_2, \ldots, C_K$, using a standard K-means algorithm. We have selected the k-means clustering [21] because it is a fast and easy to implement algorithm and also it is really fast. This algorithm allows partitioning the population of functions into k clusters, where the functions that belong to the same cluster have similar values in the training set. In this algorithm, the cluster centroid is defined as the mean data vector average over all items in the cluster. The number of clusters must be pre-assigned and the initial classification is randomly given. After that, the L–M algorithm (local search process) is applied to the best individual of each cluster. The optimized individuals for all clusters are included in a set C (local optimum set).

The combined clustering and local search process is applied dynamically every G_{t} generations and in the final one, where G_{t} must be fixed by the user. The individuals obtained with the local search in each cluster are included (stored) in the set C (dynamic local optimum set). The final solution is the best individual among the local optimums of set C (see Fig. 4).

4 Experiments and results

In this section, the configuration and the results obtained during the evaluation of the previously presented methodologies in a real wind park are presented. First, the data are described, and the parameter values of the algorithms are included. Then, the results obtained are analyzed, and some conclusions are drawn from one of the best models obtained.

4.1 Input data and configuration of the tested algorithms

As part of the training process of the proposed product unit neural network, the input variables must be selected with care. The selection involves choosing which variables are going to be used from the MM5 results, and choosing which grid points surrounding the park are considered.

Fig. 4 Pseudo-code for the dynamic hybrid evolutionary programming with clustering (*DHEPC*) algorithm

1. Generate initial population B of solutions, where each solution is a PUNN with random structure and weights
2. $g \leftarrow 0$
3. While $g < 400$ **do**:
   (a) Apply the basic EP iteration: steps 2.(a) to 2.(g), Figure 3.
   (b) Every G_t generations:
      – Apply clustering process and generate K clusters from the best $(100 \cdot \tilde{s})\%$ of individuals of B.
      – Apply L-M local search to the best individual of each cluster.
      – Add the optimized individuals to C.
   (c) $g \leftarrow g + 1$
   (d) **if** for 10 generations there was no improvement either in the average performance of the best 20% of the population or in the fitness of the best individual **then** $g \leftarrow 400$
4. End while
5. Apply one last time step 3.(b).
6. Return the best individual of C.

This selection has been carried out using the following process: first, as we have mentioned before, the MM5 model produces forecasts in specific points in a 15 km × 15 km grid, at several heights. Also, recall that the mesoscalar prediction of the wind produces correlated variables in the different points of forecasting. Thus, the idea is to look for grid points nearest to the park that present the highest correlation coefficients between the MM5 wind speed series and the mean hourly wind speed at the wind park for a given period of time. The average hourly wind speed was calculated taking the whole set of turbines integrating the park. Since these series are highly correlated with each other, the average wind speed is a good measure to retain those MM5 grid points showing the closest wind speed trends compared with those found at the park height. Since the four grid points surrounding the park were those with the highest correlation coefficients, we retained the two of them that presented the minimum correlation coefficient among them, in order to keep as much information as possible regarding wind variability. After several analysis of different sets of variables, we obtained that the following 6 inputs provided the best set of prediction variables in the neural network (note that including more inputs did not improve the performance of the neural network prediction capacity): the wind speed series in two grid points surrounding the park (s_1 and s_2), the wind direction (d), and temperature (T) in one of the previous points. Note that all these data are collected from the MM5 results at a given height, approximately equal for all the turbines, considering the orography of the park. The input variables are completed using two temporal series to obtain a measure of the solar cycle, strongly related to atmospheric circulation. We use the following equations to express the solar cycle:

$$H_1 = \sin\left(H \cdot \frac{2\pi}{24}\right) \tag{11}$$

$$H_2 = \cos\left(H \cdot \frac{2\pi}{24}\right) \tag{12}$$

where $H = [0, 23]$ is an integer vector. Note that (11) and (12) express the solar cycle during 24 h.

The hybrid forecasting system of this paper, including the product unit neural network, has been applied to the short-term wind speed prediction of a set of turbines in a wind park located at Albacete, a southeastern province of Spain. This park is known as "La Fuensanta" wind park. "La Fuensanta" is a medium size wind park, with 33 wind turbines of 1.5 MW of nominal power each. Specifically, we have carried out the wind forecasting for turbines 1, 7, 15, 21, 27, and 33, located in different points of the park. Real data from January 2006 to June 2006 at "La Fuensanta" have been obtained from anemometers situated in each wind turbine, and used in this study. On the other hand, we have collected the corresponding data from the global model considered, the data from atmospheric soundings and the *metar* data from that date. The MM5 model has been applied to them. The set of parameterizations chosen for the atmospheric processes previously listed comprise the Grell cumulus formation, the Blackardar parametrization for the planetary boundary layer, the Goodard Explicit Moisture for phase transitions, and the RRTM longwave scheme for radiation processes. Further details regarding these schemes may be found in [19]. This selection is based on previous analyses performed running the MM5 with different sets of parameterizations. The results obtained by correlating the different outputs of MM5 with wind data for "La Fuensanta" show that this set up for the MM5 model produced the highest correlation coefficients. For the sake of simplicity, these results are not shown in this paper. Finally, the output of the MM5 model run with this set of parameterizations is taken as the input data for the neural network. Note that some of these data may be not useful if they coincide with problems in the

wind turbine (stops due to excessive wind, technical stops, mechanical problems, etc). Note that, in these cases, no output at the wind turbine is available for the corresponding input data. After this filtering process, the experimental design was conducted using a holdout cross-validation procedure, with 30 runs and approximately 75% of instances for the training set and 25% for the test set. Then, we construct training sets of 6273, 5768, 6284, 6284, 6049, and 6144 samples, and test sets of 1569, 1443, 1572, 1572, 1513, and 1537 samples for turbines 1, 7, 15, 21, 27, 33 respectively. Recall that each sample consists of two values of wind speed at two points in the park, one value of wind direction in one of the points, one value of temperature at one of the points, and the two series of solar cycle. The output of the network is the corresponding value of the mean hourly wind speed at each turbine of the wind park. Part of the statistical data used in the modeling process has been included in Table 1 to better analyze and understand the nature of the problem.

Regarding the configuration of the HEP algorithms tested in this paper, all of them were implemented in language C++ and were run on a PC Pentium IV compatible computer. The principal parameters used in the algorithms can be seen in Table 2. The stopping criterion was reached whenever one of the following two conditions was fulfilled: (i) The algorithm achieved a given generation; and (ii) for 10 generations, there was no improvement either in the average performance of the best 20% of the population or in the fitness of the best individual. An analysis of variance was used for adjusting these meta-parameters. The analysis had also shown that the algorithm was quite robust to the modification of the values of the parameters within reasonable ranges. The use of an EA, which dynamically adapts to the problem evaluated, results in a performance, which is negligibly affected by minor changes in these parameters.

Table 1 Part of the statistical data used in modeling, corresponding to turbine # 1

Sample #	Training fragment						
	Inputs (MM5 data)						Output
	s_1	s_2	d	T	H_1	H_2	W_s
1	3.76	4.69	160.34	291.98	0.00	1.00	4.24
2	3.65	4.60	163.16	291.80	0.26	0.97	6.05
3	3.54	4.55	166.09	291.73	0.50	0.87	6.72
4	3.44	4.49	168.92	291.65	0.71	0.71	5.74
5	3.37	4.41	171.33	291.43	0.87	0.50	7.18
6	3.14	4.23	174.34	291.34	0.97	0.26	5.51
⋮							
6272	4.07	3.73	3.1	284.47	0.71	−0.71	3.74
6273	4.06	3.74	12.52	285.83	0.5	−0.87	3.63
	Test fragment						
1	3.80	3.45	19.71	287.49	0.26	−0.97	3.81
2	3.50	3.17	26.20	288.71	−0.00	−1.00	2.87
3	3.44	3.18	30.12	289.57	−0.26	−0.97	3.94
4	3.69	3.48	28.47	290.37	−0.50	−0.87	3.93
5	4.00	3.90	23.72	291.30	−0.71	−0.71	4.27
6	4.32	4.20	19.05	291.69	−0.87	−0.50	3.48
⋮							
1568	2.82	4.70	187.17	295.75	−0.50	0.87	3.21
1569	2.77	4.36	218.39	295.85	−0.26	0.97	7.07

The performance of the algorithms was tested using various network topologies that were run 30 times. The accuracy of each model was assessed in terms of the standard error of prediction (*SEP*) for the results obtained for the generalization set. The *SEP* was calculated as:

$$SEP = \frac{100}{\bar{W}} \sqrt{\frac{1}{n} \sum_{i=1}^{n} (W_i - \hat{W}_i)^2}; \quad SEP = \frac{100}{\bar{W}} \sqrt{MSE} \tag{13}$$

where W_i and $\hat{W}_i$ are the experimental and expected values for wind speed, $\bar{W}$ is the mean of the experimental values of the generalization set, and n is the number of patterns used for the generalization set. Finally, also the mean square error (*MSE*) obtained with each proposed algorithm is calculated in order to show their performance.

4.2 Results

Table 3 summarizes the performance of the different proposed approaches to the prediction problem in each

Table 2 Set of parameters common to all the experiments

Evolutionary algorithm's parameters	
Population size, N_P	1,000
Maximum number of generations	400
Minimum number of nodes, $M_{\min}$	3
Maximum number of nodes, $M_{\max}$	6
Exponents interval $[-M, M]$	$[-5, 5]$
Coefficients interval $[-L, L]$	$[-5, 5]$
Parametric mutation's parameters	
$\alpha_1(0) = 1$, $\alpha_2(0) = 0.5$, $\beta = 0.1$, $\rho = 10$	
Structural mutation's parameters	
IntervalNode addition $[\Delta_{MIN}, \Delta_{MAX}]$	[1, 2]
Node deletion $[\Delta_{MIN}, \Delta_{MAX}]$	[1, 2]
Connection addition $[\Delta_{MIN}, \Delta_{MAX}]$	[1, 6]
Connection deletion $[\Delta_{MIN}, \Delta_{MAX}]$	[1, 6]
Hybrid algorithm's parameters	
$K = 4, \tilde{s} = 0.25, G_t = 200$	

Table 3 Results obtained with a PUNN network as regressor in the complete wind speed forecasting model

Turbine #	1	7	15	21	27	33
Error	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD
SEP_{EP}	38.75 ± 0.37	38.64 ± 0.40	39.78 ± 0.57	39.32 ± 0.41	35.66 ± 0.28	35.92 ± 0.34
SEP_{HEPF}	38.61 ± 0.55	38.45 ± 0.47	39.81 ± 0.47	38.83 ± 1.21	35.36 ± 0.38	35.85 ± 0.47
SEP_{DHEPC}	38.58 ± 0.56	38.44 ± 0.49	40.02 ± 0.66	38.86 ± 1.22	35.36 ± 0.32	35.89 ± 0.49
MSE_{EP}	5.31 ± 0.10	4.09 ± 0.08	5.22 ± 0.15	4.96 ± 0.10	4.22 ± 0.07	3.63 ± 0.068
MSE_{HEPF}	5.27 ± 0.15	4.05 ± 0.10	5.23 ± 0.12	4.84 ± 0.32	4.15 ± 0.09	3.62 ± 0.09
MSE_{DHEPC}	5.26 ± 0.15	4.05 ± 0.10	5.28 ± 0.18	4.85 ± 0.32	4.15 ± 0.08	3.63 ± 0.10
Links	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD
$\#links_{EP}$	19.80 ± 4.16	19.53 ± 3.51	27.10 ± 2.82	25.30 ± 2.42	26.17 ± 2.67	25.33 ± 2.43
$\#links_{DHEPC}$	20.53 ± 3.21	20.57 ± 3.52	27.67 ± 2.96	25.47 ± 2.83	26.43 ± 3.19	26.30 ± 3.21
$MSE_{Best(DHEPC)}$	5.02	3.91	4.91	4.59	3.99	3.44
$\#nodes_{Best(DHEPC)}$	5 nodes	6 nodes	6 nodes	6 nodes	6 nodes	6 nodes

The results are shown with in terms of the different training methods used

considered turbine of the park. The table shows the mean and standard deviation (SD) values of *SEP* and *MSE* (over 30 runs of the algorithms), and also the errors of the best models obtained are displayed in the last rows of the table. Similar results have been included for the number of links or connections (#links) of the obtained models. As can be seen, the PUNN with DHEPC training approach outperforms (in terms of mean *SEP* and *MSE*) the other implemented training algorithms (EP and HEPF).

In order to have a further validation of the results, a comparison with a multi-layer perceptron (MLP) and radial basis function (RBF) neural network is carried out. The MLP described in [8] is used to make this comparison. The RBF neural network is a Gaussian RBF network, deriving the centers and width of hidden units using *k*-means and combining the outputs obtained from the hidden layer using a supervised linear regression.[1] The test results of the MLP and the RBF neural network with different number of neurons in the hidden layer are shown in Table 4. Note that the proposed PUNN with the DHEPC training is able to improve the results of both methods in all the turbines considered. Note that the best MLP is usually obtained with 12 or 14 neurons in the hidden layer and the best RBF number of neurons ranges from 9 to 11 neurons. The lowest MSE difference between the best MLP and the EPUNN model is 0.2 (Turbine 15), and the largest MSE difference is 0.59, obtained for Turbine 1. For the RBF, the lowest MSE difference is 0.79 (Turbine 1), and the largest MSE difference is 1.51, obtained for Turbine 21.

The interest of the proposed PUNNs is not only that they improve the results of other approaches (MLP and RBF), as shown before, but also that we can obtain a (non-linear) interpretable model of the system. In this case, we show and interpret the model obtained for Turbine 1 of the park.

Table 4 Test results obtained (MSE) with a multi-layer perceptron (MLP) and a radial basis function (RBF) neural network as regressors compared with the results obtained by the best EPUNN found with the DHEPC algorithm

Turbine #	MLP (# NHL)							EPUNN
	9	10	11	12	13	14	15	
1	5.70	5.64	5.62	5.67	5.64	5.61	5.74	5.02
7	4.27	10.05	7.14	4.25	4.28	4.27	7.48	3.91
15	5.15	5.13	8.09	5.13	5.13	5.11	5.12	4.91
21	5.09	8.33	8.35	5.03	5.06	5.03	5.03	4.59
27	4.44	4.46	4.46	4.42	4.43	4.44	4.45	3.99
33	3.91	3.90	3.90	3.93	3.91	3.92	6.48	3.44
	RBF (# NHL)							
1	5.95	5.81	5.99	6.20	6.52	6.89	7.02	5.02
7	5.32	5.42	5.30	5.33	5.35	5.48	5.48	3.91
15	6.31	6.41	6.34	6.53	6.64	7.21	7.27	4.91
21	6.18	6.10	6.39	6.37	6.65	7.10	7.15	4.59
27	5.18	5.04	5.15	5.53	5.52	6.22	6.26	3.99
33	4.47	4.46	4.61	4.86	4.87	5.19	5.21	3.44

NHL stands for the number of neurons in the hidden layer

4.2.1 Analysis of the best PUNN models obtained

The best PUNN model obtained for Turbine 1 (the one in which the EPUNN model performs much different than the MLP) is shown in Table 5. A graphical representation has been included in Fig. 2 for an easier analysis. Note that the algorithm selects the most important inputs thanks to the

[1] For this comparison, we have considered the WEKA machine learning workbench with the corresponding RBFNetwork algorithm (http://www.cs.waikato.ac.nz/ml/weka/).

Table 5 Best model obtained with the DHEPC approach for wind Turbine 1

Best model for Turbine #1
$\hat{W}_s = 0.471 - 0.669 \cdot PU_1 + 0.511 \cdot PU_2 - 4.010 \cdot PU_3 - 5.722 \cdot PU_4 + 4.998 \cdot PU_5$
$PU_1 = T^{0.296} \cdot H_1^{0.010}$
$PU_2 = s_1^{0.075} \cdot d^{-0.062} \cdot T^{0.580} \cdot H_2^{0.057}$
$PU_3 = s_1^{-0.061} \cdot s_2^{2.818} \cdot d^{2.612} \cdot H_1^{0.017}$
$PU_4 = s_1^{2.186} \cdot s_2^{-0.823} \cdot d^{16.493} \cdot H_2^{-0.018}$
$PU_5 = s_1^{1.018} \cdot s_2^{1.030} \cdot d^{2.388} \cdot H_1^{0.034} \cdot H_2^{0.014}$
$\#\text{nodes} = 5,\ \#\text{links} = 25$
$SEP_G = 37.67,\ MSE_G = 5.02$
$s_1, s_2, d, T, H_1, H_2 \in [0.1, 0.9]$

Variables in the model (see Sect. 4.1) are s_1: wind speed in point 1, s_2: wind speed in point 2, d: wind direction, T: temperature, H_1 and H_2: components of the solar cycle

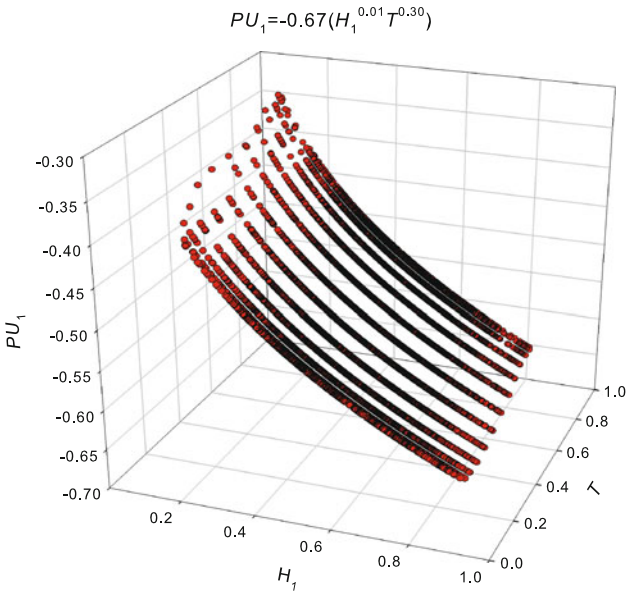

Fig. 5 Relation between variables T (temperature), H_1 (solar cycle) and product unit PU_1

corresponding structural mutators (add connection and remove connection). It is easy to see that the most important product units are PU_1 and PU_2, in this order. The variables that most influence this turbine are T and H_1 in PU_1 and s_1, d and H_2 in PU_2. The former is inversely related to the model output ($\hat{W}_s$), and the latter is directly related to $\hat{W}_s$. Thus, an increment in T means a strong increment of PU_1, whereas, on the other hand, a strong increment of H_1 implies a small increment in PU_1. This means that PU_1 reaches a maximum value for T large, and H_1 small, see Fig. 5. In addition, note that the larger is PU_1, the smaller $\hat{W}_s$.

Mathematically, PU_1 is modeled as follows:

$$PU_1 = (T)^{0.296} \cdot (H_1)^{0.010} \tag{14}$$

and PU_2 is modelled as:

$$PU_2 = (s_1)^{0.075} \cdot (d)^{-0.062} \cdot (T)^{0.580} \cdot (H_2)^{0.057} \tag{15}$$

Figure 5 shows the construction of product unit PU_1 as a function of variables T and H_1. We can observe that there is a kind of interaction between both variables, as expected (T stands for the temperature in a point of the wind park and H_1 stands for the first component of the solar solar cycle).

A different analysis can be carried out on the linearity between the real values of W_s and the most important product units PU_1 and PU_2: if we consider only these two product units (PU_1 and PU_2 in this case), we would have a regression hyper-plane given by:

$$\hat{W}_s(PU) = 0.471 - 0.669 \cdot PU_1 + 0.511 \cdot PU_2 \tag{16}$$

instead of the initial equation given in Table 5. Note that we could construct a much simpler linear regressor considering only PU_1 and PU_2, though this model would be less efficient than the original one.

Finally, note that we have obtained a model similar to the one described in this paper for each turbine considered in the study. The description of the models for all the turbines is quite similar to the process carried out before in Turbine 1. Instead of the complete description of all the models, we summarize the results in Table 6, where we show the most important product units for each turbine, with the variables with a higher influence in the wind speed prediction. Note that the set of most important variables in the wind speed prediction depends on the turbine considered. This is expected, since we have chosen turbines in different parts of the wind farm, far away one from the others. It is interesting to note that in the majority of the turbines, the most important variables (the ones which capture the main information of wind speed prediction) are the wind speed in a given point and the solar cycle, whereas other variables such as temperature or wind direction are less important. However, these variables are also included in the complete models, so we cannot just remove them, since the model would be less accurate than the current ones.

A final experiment has been carried out in order to prove the necessity of the rest of variables. Since H_1 and H_2 appear to be the most important variables, we can study the relationship between the real output for all the training points (W_s) and these two variables (again only for Turbine 1). Figure 6 shows that the relationship between the two variables and the real output is clearly non-linear, and also, that W_s is independent of the joint value of H_1 and H_2.

Table 6 Most important product units and variables in each considered turbine

Turbine	Order	
	Product units	Variables
1	$PU_1^{(-)}\ PU_2^{(+)}$	$H_1^{(-)}\ H_2^{(+)}\ d^{(-)}\ s_1^{(+)}$
7	$PU_4^{(-)}\ PU_5^{(+)}$	$d^{(-)}\ H_2^{(+)}$
15	$PU_3^{(+)}\ PU_5^{(-)}$	$H_2^{(+)}\ s_2^{(-)}$
21	$PU_4^{(-)}\ PU_5^{(+)}\ PU_3^{(-)}$	$s_2^{(+)}\ H_2^{(+)}\ T^{(\sim)}\ s_1^{(-)}\ H_1^{(-)}$
27	$PU_5^{(-)}\ PU_2^{(-)}\ PU_4^{(+)}$	$H_2^{(\sim)}\ s_1^{(\sim)}\ H_1^{(-)}$
33	$PU_1^{(+)}\ PU_2^{(-)}\ PU_4^{(+)}$	$H_2^{(+)}\ H_1^{(\sim)}\ d^{(\sim)}$
	$s_1, s_2, d, T, H_1, H_2 \in [0.1, 0.9]$	

(+): directly proportional to wind speed; (−): inversely proportional to wind speed; (∼): not clearly related to wind speed; s_1: speed in the first point; s_2: speed in the second point; d: direction in the first point; T: temperature in the second point; H_1: first daily solar cycle component; H_2: second daily solar cycle component

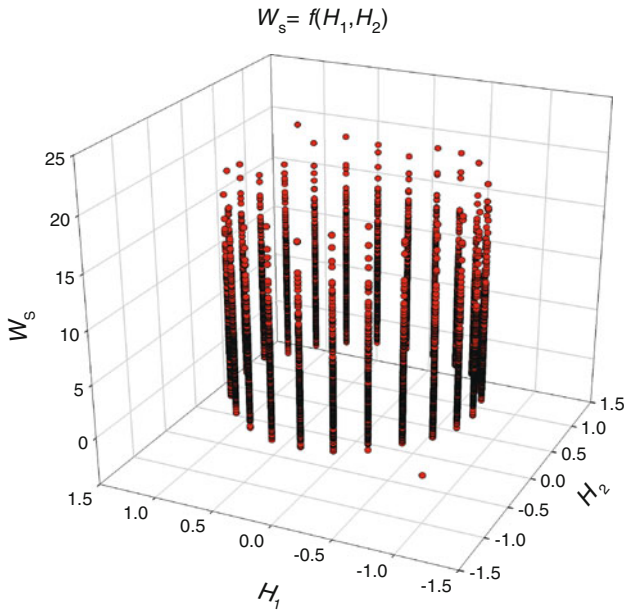

Fig. 6 Relation between real values of W_s and variables H_1 and H_2 (solar cycle)

Because of this reason, it is clear that, although these two variables are very important, a linear regressor trying to predict the output only from a linear combination of both values would not be enough for this case, and proper non-linear model (as PUNNs are) can perform a better prediction.

5 Conclusions

This study demonstrates the capability of evolutionary product unit neural networks (EPUNNs) to perform accurate short-term wind prediction. A previous system [8] has been modified by replacing a previous multi-layer perceptron by an evolutionary product unit neural network (EPUNN). Moreover, three different EPUNN algorithms (EP, HEPF, and DHEPC) have been evaluated in PUNNs training for the wind speed prediction problem in several different turbines of a wind farm in Spain. The statistical results show that DHEPC yields better results than the other proposed approaches, with lower MSE mean and standard deviation. When compared with the MLP and RBF neural networks, the best EPUNNs obtained a lower MSE with a lower number of coefficients.

The low complexity (in terms of number of coefficients) of EPUNNs allowed the interpretation of the relation between the wind speed and the different variables selected as inputs. This kind of interpretation is not usually considered in previous wind speed prediction research works, specially when using MLPs, since the high number of coefficients and neurons make difficult to isolate the non-linear effects of the variables in the predicted response. In this paper, we have shown the study of the best model for one turbine of the wind farm, drawing interesting conclusions on how the different variables of the models affect the final results obtained.

Acknowledgments This work has been partially supported by Spanish Ministry of Industry, Tourism and Trading, under an Avanza 2 project, number TSI-020100-2010-663, by the TIN 2008-06681-C06-03 project of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds and by the P08-TIC-3745 project of the "Junta de Andalucía" (Spain).

References

1. Kusiak A, Zheng H, Song Z (2009) Wind farm power prediction: a data-mining approach. Wind Energy 12(3):275–293
2. Barthelmie RJ, Murray F, Pryor SC (2008) The economic benefit of short-term forecasting for wind energy in the UK electricity market. Energy Policy 36(5):1687–1696
3. Mabel MC, Fernández E (2008) Analysis of wind power generation and prediction using ANN: a case study. Renew Energy 33(5):986–992
4. Louka P, Galanis G, Siebert N, Kariniotakis G, Katsafados P, Pytharoulis I, Kallos G (2008) Improvements in wind speed forecasts for wind power prediction purposes using Kalman filtering. J Wind Eng Ind Aerodyn 96(12):2348–2362
5. Hocaoglu FO, Oysal Y, Kurban M (2009) "Missing wind data forecasting with adaptive neuro-fuzzy inference system. Neural Comput Appl 18(3):207–212
6. Goh SL, Chen M, Popovic DH, Aihara K, Obradovic D, Mandic DP (2006) Complex-valued forecasting of wind profile. Renew Energy 31:1733–1750
7. Barbounis TG, Theocharis JB (2007) Locally recurrent neural networks for wind speed prediction using spatial correlation. Inf Sci 177:5775–5797
8. Salcedo-Sanz S, Pérez-Bellido ÁM, Ortiz-García EG, Portilla-Figueras A, Prieto L, Paredes D (2009) Hybridizing the fifth generation mesoscale model with artificial neural networks for

short-term wind speed prediction. Renew Energy 34(6): 1451–1457
9. Landberg L, Giebel G, Nielsen HA, Nielsen TS, Madsen H (2003) Short-term prediction—an overview. Wind Energy 6(3):273–280
10. Li S, Wunsch DC, O'Hair E, Giesselmann M (2001) Comparative analysis of regression and artificial neural network models for wind turbine power curve estimation. J Solar Energy Eng 123:327–332
11. Maqsood I, Riaz-Khan M, Abraham A (2004) An ensemble of neural networks for weather forecasting. Neural Comput Appl 13(2):112–122
12. Mohandes MA, Halawani TO, Rehman S, Hussain AA (2004) Support vector machines for wind speed prediction. Renew Energy 29:939–947
13. Torres JL, García A, De Blas M, De Francisco A (2005) Forecast of hourly average wind speed with ARMA models in Navarre (Spain). Solar Energy 79:65–77
14. Durbin R, Rumelhart D (1989) Product units: a computationally powerful and biologically plausible extension to backpropagation networks. Neural Comput 1:133–142
15. Schmitt M (2001) On the complexity of computing and learning with multiplicative neural networks. Neural Comput 14:241–301
16. Martínez-Estudillo A, Martínez-Estudillo F, Hervás-Martınez C, García-Pedrajas N (2006) Evolutionary product unit based neural networks for regression. Neural Netw 19:477–486
17. Ismail A, Engelbrecht AP (2000) Global optimization algorithms for training product unit neural networks. In: Proceedings of the IEEE international conference on neural networks, Como, Italy. IEEE Press
18. Kanamitsu M, Alpert JC, Campana KA, Caplan PM, Deaven DG, Iredell M, Katz B, Pan HL, Sela J, White GH (1991) Recent changes implemented into the global forecast system at NMC. Weather Forecast 6(3):425–435
19. Dudhia J (1993) A nonhydrostatic version of the Penn State-NCAR Mesoscale Model: validation, tests and simulation of an atlantic cyclone and cold front. Mon Weather Rev 121:1493–1513
20. Martínez-Estudillo AC, Hervás-Martínez C, Martínez-Estudillo FJ, García-Pedrajas N (2006) Hybridization of evolutionary algorithms and local search by means of a clustering method. IEEE Trans Syst Man Cybern B 36(3):534–546
21. Fukunaga K (1990) Introduction to statistical pattern recognition. Academic Press, San Diego
22. Steenbeek AG, Marchiori E, Eiben AE (1998) Finding balanced graph bipartitions using a hybrid genetic algorithm. In: Proceedings of the IEEE international conference on evolutionary computation, pp 90–95
23. Houck CR, Joines JA, Kay MG (1997) Empirical investigation of the benefits of partial lamarckianism. Evol Comput 5:31–60
24. Joines JA, Kay MG (2002) In: Sarker R, Mahamurdian M, Yao X (eds) Evolutionary optimisation. Kluwer, Boston
25. Angeline P, Sauders G, Pollack J (1994) An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans Neural Netw 5:54–65
26. Yao X, Liu Y (1997) A new evolutionary system for evolving artificial neural networks. IEEE Trans Neural Netw 8(3):694–713
27. García-Pedrajas N, Hervás-Martínez C, Muñoz-Pérez J (2002) Multiobjective cooperative coevolution of artificial neural networks. Neural Netw 15(10):1255–1274
28. Gutiérrez PA, Hervás C, Carbonero M, Fernández JC (2009) Combined projection and kernel basis functions for classification in evolutionary neural networks. Neurocomputing 72(13-15):2731–2742
29. Houck CR, Joines JA, Kay MG (1996) Comparison of genetic algorithms, random restart and two-opt switching for solving large location-allocation problems. Comput Oper Res 23:587–596