Designing Beta Basis Function Neural Network for Optimization Using Artificial Bee Colony (ABC)

Habib Dhahri, *Student Member*, IEEE, Adel. M. Alimi, *Senior Member*, IEEE,
Ajith Abraham, *Senior Member*

***Abstract*— This paper presents an application of swarm intelligence technique namely Artificial Bee Colony (ABC) to design the design of the Beta Basis Function Neural Networks (BBFNN). The focus of this research is to investigate the new population meta-heuristic to optimize the Beta neural networks parameters. The proposed algorithm is used for the prediction of benchmark problems. Simulation examples are also given to compare the effectiveness of the model with the other known methods in the literature. Empirical results reveal that the proposed ABC-BBFNN have impressive generalization ability.**

I. INTRODUCTION

The neural networks have been effectively applied in many areas, such as time series prediction [1,2,3], pattern recognition [4], approximation function [5,6,28], etc. Among the artificial neural networks, the beta basis function neural networks (BBFNN), represents an interesting alternative in which we can approximate any function [7]. The BBFNN network is a three –layer feed-forward networks that generally uses a linear transfer function for the output units and a non-linear transfer function (the beta function) for the hidden units. In spite of a number of advantages of BBFNN such as better approximation capabilities [8], faster learning algorithms and simple network topologies; especially the determination of the optimal number of hidden nodes is the most critical task. The development of BBFNN still involves difficulties in optimizing the topology of the network structure (the number of nodes). Today, hybridization in soft computing is becoming a promising research field of computational intelligence focusing on synergistic combinations of multiples soft computing methodologies an intelligent system. In order to overcome the soft computing method [9-11], the investigation of hybrid approaches will be necessary. In particular, in order to overcome the challenge in developing the neural network, the evolutionary algorithm is applied to optimize the structure of the neural network system. There are several works that deal the problem of neural network design [12-15]. The applying of evolutionary algorithms to construct neural nets is also well known in the literature. The most representative algorithms include genetic algorithms (GA) [16][46], particle swarm optimization (PSO) [17,18,19], the differential evolution (DE) [2,3,17,20,21,22,23], flexible neural trees [47,48]. In [24], ABC is used to optimize a large set of numerical test functions and the results produced by ABC algorithm are compared with the results obtained by genetic algorithm, particle swarm optimization algorithm, differential evolution algorithm and evolution strategies. Results show that the performance of the ABC is better than or similar to those of other population-based algorithms with the advantage of employing fewer control parameters.

In this paper, we investigate the advantages of the artificial bee colony (ABC) optimization algorithm to the population-based metaheuristic on training the beta basis function neural networks (BBFNN) for the prediction of benchmark problems which are widely used in the machine learning community. The performance of the ABC algorithm is compared with other well-known conventional and evolutionary algorithms. The results indicate that that ABC algorithm can efficiently be used on training beta basis function neural networks.

The paper is organized as follows: in Section 2 we briefly present the basics of BBFNN. In Section 3, we explain the fundamental concept of ABC algorithm and how ABC algorithm is used to design of BBFNN. In Section 4, the experimental results using benchmark problems are given. Finally, in Section 5 we present the conclusions of the work.

II. BBFNN NETWORK

In this Section, we introduce the beta basis functions neural network that will be used in the remainder of this paper. The BBFNN usually consists of three layers the input layer, the BBF layer (hidden layer) and the output layer. The input layer simply transfers the input vector $x = [x_1, x_2, ..., x_n]^T$ through scalar weights to the next layer. Thus the whole input vector appears to each neuron in the hidden layer. Each hidden nodes perform the beta basis function over the incoming vector that appears at the input of each BBFNN neuron. The output layer yields a vector $y = [y_1, y_2, ..., y_m]^T$ for m outputs by linear combination of the outputs of the hidden nodes to produce the final output.

H. Dhahri is with University of Sfax, ENIS, Department of Electrical BP W-3038, Sfax, Tunisia, Tel +216-74-274-088,
e-mail: habib.dhahri@ieee.org.
A.M. Alimi is with University of Sfax, ENIS, Department of Electrical BP W-3038, Sfax, Tunisia; Tel +216-74-274-088,
Fax. +216-74-275-595, e-mail adel.alimi@ieee.org.
A. Abraham is with Faculty of Electrical Engineering and computer science, Technical University of Ostrava, Czech Republic,
Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence (SNIRE), WA, USA,
e-mail ajith.abraham@ieee.org

Figure 2 presents the structure of a single output of BBF network; the network output can be obtained by

$$y = f(x) = \sum_{i=1}^{n} w_i B_i(x, c, \sigma, p, q), 1 \le i \le n \quad (1)$$

Besides the centre c, the beta basis function may also present a width parameter σ, which can be seen as a scale factor for the distance (x-c) and the parameter forms p and q. The BBF network can be regarded as feed-forward neural network with a single layer of hidden units, whose responses are the outputs of the beta basis functions. The Fig.1 shows the effect of parameters forms to the Beta function .The latter $B_i(x, c_i, \sigma_i, p_i, q_i)$, i =1,…,n, is defined by:

$$\beta(x) = \begin{cases} \left[1 + \frac{(p+q)(x-c)}{\sigma p}\right]^p \left[1 - \frac{(p+q)(c-x)}{\sigma q}\right]^q \\ \quad if \quad x \in]x_0, x_1[\\ 0 \quad else \end{cases} \quad (2)$$

III.

Where p > 0, q > 0, x_0, x_1 are the real parameters, such as $x_0 < x_1$ and

$$c = \frac{px_1 + qx_0}{p+q} \quad (3)$$

In the multi-dimensional case, the beta function is defined by

$$\beta(c, \sigma, p, q)(x) = \prod_{i=1}^{i=d} \beta_i(c_i, \sigma_i, p_i, q_i)(x) \quad (4)$$

Where d is the dimension of the Beta kernel.

The BBF neural network is usually trained to map a vector $x_k \in \mathbb{R}^n$ in to vector $y_k \in \mathbb{R}^{no}$ where the pairs $(x_k y_k), 1 \le k \le M$ from the training set. If this mapping is viewed as a function in the input space $\mathbb{R}^n$, learning can be seen as function approximation problem. According to this point of view, learning is equivalent to finding the surface in a multidimensional space that provides the best fit to the training data. Generalization is therefore synonymous with interpolation between the data points along the constraint surface generated by the fitting procedure as the optimum approximation to this mapping.

Alimi [7] investigated the use of the beta basis function in the design of neural network as activation functions in artificial neural networks. In [8], the authors proved that BBF networks with one hidden layer are capable of universal approximation. Nevertheless, the BBF networks are capable of approximating arbitrarily well any function; also have the best approximation property.

The performance of the BBF neural network depends to the number of units of beta basis functions, their shapes, the parameters forms, and the method used to determine the associative weight matrix. Haykin [25] classified the existing learning strategies for neural network as follows:1) learning with a fixed number of units and centers selected randomly from the training data; 2) supervised learning for the selection of the centers of the network; and 3) unsupervised learning for the selection of the fixed number of units. In this paper, we used the second strategy.

One of the main problems related to the development of neural network based system is the application of suitable learning algorithm to adjust the network parameters. The BBF network presents the following adjustable parameters: the position of BBFs centers c_i ,the widths σ_i of the BBFs, the form parameters of the BBFs p_i and q_i and the output weights w_i .

There are a number of proposals on how to define these parameters in the literature. One first idea is to fix the number of nodes and use a gradient descent method to adjust the parameters [26], in a manner very similar to the error back-propagation algorithm, often used with MLPs. Nevertheless, training the BBF network in such a way seems somewhat wasteful. There are several interesting approaches that exploit this potential. Although slightly different, all of them share the same idea: the definition of the hidden layer is considered as the major task, since the output weights can be computed according to linear optimization techniques [26]. In [27] the authors used the constructive method that allows BBF neural network to grow by inserting new units in the feature space where the mapping needs more details. In [5,6,17], the major task considered in these works is to optimize the beta parameters with a fixed number of nodes. In the proposed work, we use the second strategy of Haykin.

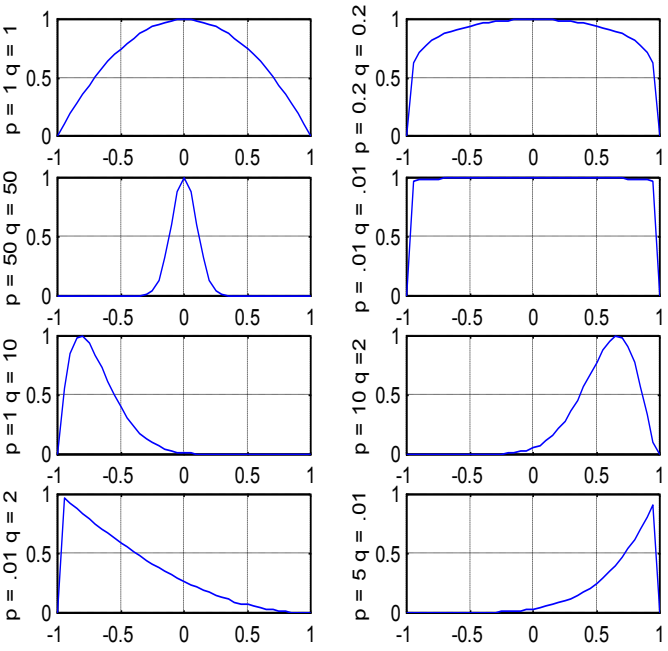


Fig.1. the Beta plot in one dimension

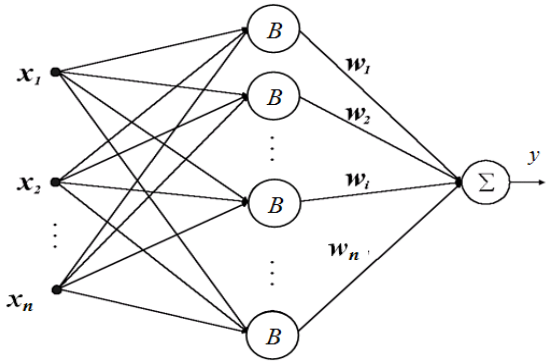


Fig. 2. The beta architecture

III. Artificial Bee Colony Algorithm

ABC is a swarm intelligent optimization algorithm based on the metaphor foraging behavior of honey bee swarm, proposed by Karaboga in 2005 [29]. A Bee Colony can be considered as swarm whose individual social agents are bees. The exchange of information among bees leads to the formation of tuned collective knowledge. Virtually the bee colony consists of a single "queen bee," a few hundred drones (males), and tens of thousands of workers (non-reproductive females).

In the ABC algorithm, each food source is a possible solution for the problem under consideration and the nectar amount of a food source represents the quality of the solution represented by the fitness value. The number of food sources is same as the number of employed bees and there is exactly one employed bee for every food source. In the ABC model, the colony consists of three groups of bees: employed bees, onlookers and scouts. It is assumed that there is only one artificial employed bee for each food source. In other words, the number of employed bees in the colony is equal to the number of food sources around the hive. Employed bees go to their food source and come back to hive and dance on this area. The employed bee whose food source has been abandoned becomes a scout and starts to search for finding a new food source. Onlookers watch the dances of employed bees and choose food sources depending on dances.

The number of the employed bees is equal to the number of solutions in the population.

The general scheme of the ABC algorithm is as follows:

Initialize Population
repeat
Place the employed bees on their food sources
Place the onlooker bees on the food sources depending on their nectar amounts
Send the scouts to the search area for discovering new food sources
Memorize the best food source found so far
until requirements are met

A. Population initialization

As with all swarm intelligent techniques, ABC works with a population of solutions, not with a single solution for the optimization problem. In order to establish a starting point for optimum seeking, all employed bees are associated with the food. A randomly distributed initial population (food source positions) is generated. Often there is no more available knowledge about the location of a global optimum than the boundaries of the problem variables. In this case, a natural way to initialize the population, P (initial population) is to seed it with random values within the given the lower and upper bound:

$$x_{mi} = l_i + rand(0,1) \times (u_i - l_i) \qquad (5)$$

where l_i and u_i are the lower and upper bound of the parameter x_{mi}, respectively.

B. Initialization of Bee Phase (Employed Bee)

Each employed bee search a food source v_m having more nectar in the neighborhood of its current food source and evaluates its nectar amount (fitness). The employed bee saved the best food x_i in the neighborhood of its present position by using:

$$v_{mi} = x_{mi} + \phi_{mi} \times (x_{mi} - x_{ki}) \qquad (6)$$

Where x_k is a randomly selected food source, i is a randomly chosen parameter index and ϕ_{mi} is a random number within the range $[-a,a]$. After producing the new food source v_m, its fitness is calculated and a greedy selection is applied between v_m and x_m

C. Onlooker Bee Phase

Onlooker bee chooses a food source depending on the probability values calculated using the fitness values provided by employed bees. For this purpose, a fitness based selection technique can be used, such as the roulette wheel selection method [30]

The probability value p_m with which x_m is chosen by an onlooker bee can be calculated by using the expression given in the following equation

$$p_m = \frac{fit_m}{\sum_{k=1}^{Fs} fit_k} \qquad (7)$$

Where fit_m is the nectar amount of the of the m^{th} food source. The fitness value of the solution, $fit_m(x_m)$, might be calculated for minimization problems using the following formula:

$$fit_m = \frac{1}{1+f(x_m)} \qquad (8)$$

After a food source x_m for an onlooker bee is probabilistically chosen, a neighborhood source v_m is determined by using equation 6, and its fitness value is computed. As in the employed bees phase, a greedy selection is applied between v_m and x_m. Hence, more onlookers are recruited to richer sources and positive feedback behavior appears.

D. Scout Bee Phase

The unemployed bees that choose their food sources randomly are called scouts. Employed bees whose solutions cannot be improved through a predetermined number of trials, "limit" or "abandonment criteria" herein, become scouts and their solutions are abandoned. For instance, if solution x_m has been abandoned, the new solution discovered by the scout who was the employed bee of x_m can be defined by equation 5. The pseudo code of ABC algorithm is:

---

1) Initialize the population of solution x_{ij}
2) Evaluate the population
3) While number of cycle is not reached do
4) Produce a new solution (food sources positions) y_{ij} in the neighborhood of x_{ij} using equ.6 and evaluate them.
5) Apply the greedy selection and store the best values between x_{ij} and v_{ij}
6) Calculate the probability values p_i for different solution x_i by equ .7
7) Based on the probability p_i, new solutions v_i for the onlookers are produced from the x_i
8) Apply the greedy selection and store the best values between x_{ij} and v_{ij}
9) Determine the abandoned solution (position or source) for the scout if exit and replace it with a new randomly produced solution x_i by equ.5
10) Memorize the best food source solution achieved s_i
11) Cycle =cycle+1
12) End of while

E. Encoding scheme of BBFNN networks

The ABC is used to optimize the neural parameters of the BBFNN. This approach handles the task of updating the population for neural network- neuron optimization. Each individual of the population defines a beta basis function neural network. The ABC algorithm is used to train BBFNN by adjusting the neural parameters with individuals with the same size. After a predefined number of generations, ABC returns the best individuals that represent the optimal configuration of BBFNN.

Once applying the ABC algorithm to design the BBFNN network, the main key is to encode the BBF neural network into the chromosome with an efficient approach. Here, we adopt the real coded ABC and each sequence of neural parameters (Figure 3) represents one node. Each chromosome represents a candidate BBFNN neural network. Since the weights parameters are computed by the pseudo-inverse technique, therefore it is only necessary to encode the four parameters, i.e., centers c_i , widths σ_i, and form parameters p_i and q_i which are necessary to represent the Beta form of BBF .

c_{11}	σ_{11}	p_{11}	q_{11}	….	c_{1m}	σ_{1m}	p_{1m}	q_{1m}
….	….	….	….	….	….	….	….	….
c_{n1}	σ_{n1}	p_{n1}	q_{n1}	….	c_{nm}	σ_{nm}	p_{nm}	q_{nm}

Fig. 3. the encoding scheme

IV. COMPUTATIONAL EXPERIMENTS

The developed ABC_BBFNN model is applied to three benchmark problem in order to compare is performance with existing technique. These problems are the Box–Jenkins and sunspot number time series.

A. Prediction of Box-Jenkins Time series

In this section, the Beta basis function neural network is applied to the gas furnace data prediction problem [31]. The data set was recorded from combustion process of a methane–air mixture. It is well known and frequently used as a benchmark example for testing identification and prediction algorithms. The data set consists of 296 pairs of input–output measurements. The input of this process is the gas flow rate *u(t)* and the output *y(t)* is the CO2 concentration in outlet gas. in order to make a meaningful comparison with the others work s, the inputs of the prediction model are selected *as u(t-4)* and *y(t-1)*; and the output is *y(t)*: the proposed ABC learning algorithm is employed to train the BBFNN with the first 200 input - output. The remaining 92 points are used as a test set for testing the performance. In order to remove the effects of the initial values of free parameters on the final results, 20 runs were performed with randomly set initial parameters for 1000 epochs.

The objective function used is the root mean square error (RMSE).Table 1 shows the comparison of test results of different models for Box–Jenkins data prediction problem. The comparison has been made to show the actual time-series, the output of the best ABC-BBFNN and the prediction error and the number of neurons. It is seen from the training performances that ABC-BBFNN model with 3 neurons is among the best models. The ABC-BBFNN is powerful for the Box-Jenkins process in training and testing. The proposed algorithm is trained for 1000 epochs (Figure 4). Figure 5 shows the target time series with the output of ABC-BBFNN and Figure 6 depicts the prediction of the time series.

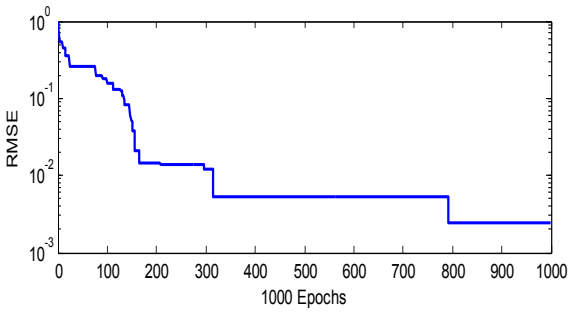
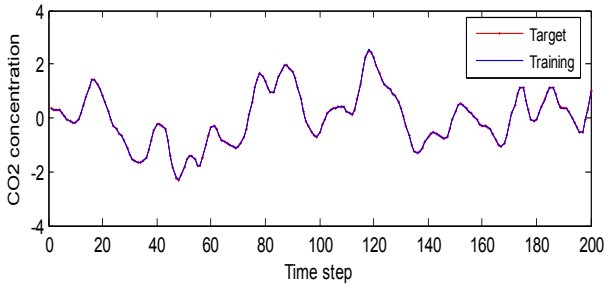


Fig.4. RMSE training of Box-Jenkins



Fg.5. ABC-BBFNN result of Box-Jenkins

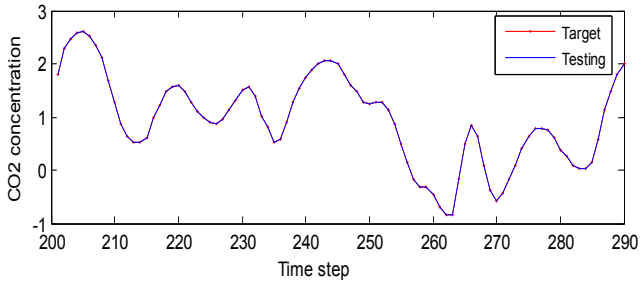


Fig.6. Test of ABC-BBFNN for Box-Jenkins

Table 1: Comparison of different models of Box-Jenkins Time Series

Model	RMSE Training	RMSE testing
Tong's model [32]	-	0.689
Pedrycz's model [33]	-	0.566
Xu's model [34]	-	0.573
FuNN model [35]	-	0.0226
HyFis model [36]	-	0.0205
Neural tree model [37]	0.0258	0.0265
WNN+ gradient [31]	0.08831	0.084
WNN+ hybrid [31]	0.08485	0.081
LLWNN+ gradient [31]	0.01581	0.01643
LLWNN+ hybrid [31]	0.01095	0.01378
Recurrent ANFIS[31]	0.006	0.019
TNFIS [40]	0.0245	0.0230
FWNN-S (2 MFs) [39]	0.01884	0.03085
FWNN-S (3 MFs) [39]	0.01880	0.02778
FWNN-R (2 MFs) [39]	0.01992	0.03171
FWNN-R (3 MFs) [39]	0.1881	0.02794
FWNN-M (2 MFs) [39]	0.0190	0.02963
FWNN-M (3 MFs) [39]	0.01963	0.02324
Our approach	0.0044	0.0049

B. Prediction of sunspot number time series

The series studied here represents the annual average number of sunspots. These numbers show the yearly average relative number of sunspot observed [41,44,,43]. The data points between 1700 and 1900 are used for the training the BBFNN and 1901 -1990 for the test set. The y(t-4),y(t-3), y(t-2) and y(t-2) are used as inputs to the ABC-BBFNN in order to predict the output y(t). The normalized mean square error NMSE is used to compare the propose algorithm with the other approaches.

Table 2 illustrates the comparison of the proposed algorithm with other models according to the training and testing error. In Figure 7, the actual output of the time series. The prediction values are illustrated in Figure 8 and Figure 9 gives the training error. As evident from Table 2, ABC-BBFNN shows again the efficiencies for the sunspot number time series.

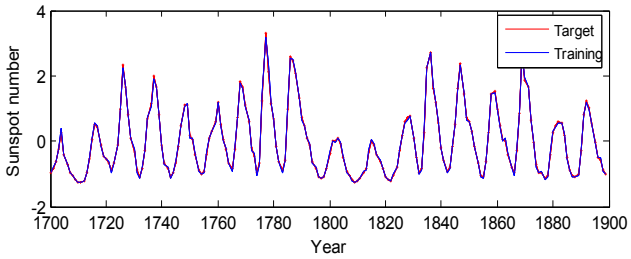


Fig.7 ABC-BBFNN result of Sunspot

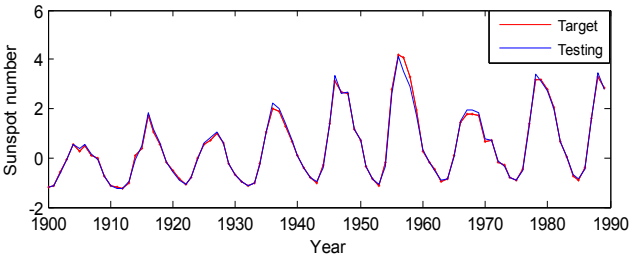


Fig.8. Test of ABC-BBFNN for Box-Jenkins

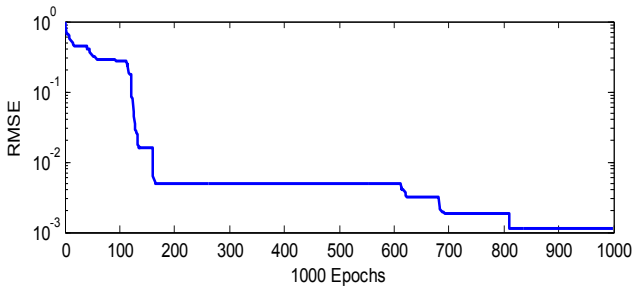


Fig.9. RMSE training of sunspot

Table 2. Comparison of different models of Sunspot Time Series prediction

Model	RMSE Training	RMSE testing 1	RMSE testing 2
Transversal Net [41]	0.0987	0.0971	0.3724
Recurrent net [41]	0.1006	0.0972	0.4361
RFNN [42]	-	0.074	0.21
ANFIS [43]	0.0550	0.1915	0.4068
FENN [44]	-	-	0.18
FWNN-S [39]	0.0895	0.1093	0.1510
FWNN-R [44]	0.0796	0.1099	0.2549
FWNN-M [44]	0.0828	0.0973	0.1988
Our approach	0.0012	0.0018	0.0044

C. *Lorenz chaotic time series prediction*

The Lorenz system is an idealized model of fluid motion between a hot surface and a cool surface. It is described by the following nonlinear ordinary differential equations

$$\dot{x} = \sigma(y - x),$$
$$\dot{y} = -y - xz + Rx, \quad (9)$$
$$\dot{z} = xy - bz,$$

The time series used in this experiment is the x-component in the Lorenz equations. The data were generated by solving the system of differential equations, that describe the Lorenz attractor, with the initial conditions of $\sigma = 10$, $r = 50$ and $b = 8/3$. The data were again normalized to take values from zero to one, before they were used as inputs to the polynomial neural networks.

The objective is to make one-step a head prediction. The prediction is based on four past values $(x(t-1), x(t-2), x(t-3), x(t-4))$ and thus the output pattern is $x(t) = f(x(t-1), x(t-2), x(t-3), x(t-4))$.

Figure 10 presents the comparison between the real time series and that predicted by the algorithm, using 4 input variables, in order to predict the value of the time series (1 step), using 4 neurons in the hidden layer. The root means square error, for this simulation was 0.076. It is important to note that other approaches appeared in the bibliography, for example, Xiang et al. [45] obtained an RMSE of 0.290. Figure.11 depicts the results of predicting the Lorenz time series.

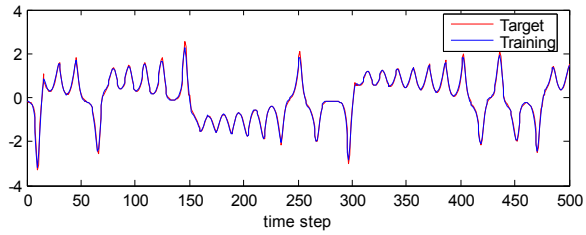


Fig.10 ABC-BBFNN result of Lorentz

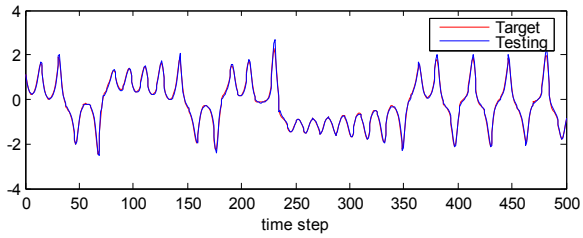


Fig.11. Test of ABC-BBFNN for Lorentz

V. CONCLUSIONS

In this paper, the beta basis function neural network is developed for the prediction of benchmark problems. The impressive generalization capability of the presented BBFNN model is derived primarily from the use of the artificial of bee colony algorithm (ABC) and the fast convergence with high precision. As evident from the experiments, the ABC-BBFNN gives the smallest training error and the testing error.

The results obtained through the sunspot, Box Jenkins also reveal that the performance of the ABC is better than or similar to those of other population-based algorithms with the advantage of employing fewer control parameters. Furthermore, in ABC, no user intervention is required. Our future work is targeted to improve the ABC to further enhance the optimal structure of BBFNN.

ACKNOWLEDGMENT

The authors would like to acknowledge the financial support of this work by grants from the General Direction of Scientific Research and Technological Renovation (DGRSRT), Tunisia, under the ARUB program 01/UR/11/02. Ajith Abraham acknowledges the support from the framework of the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 supported by Operational Programme 'Research and Development for Innovations' funded by Structural Funds of the European Union and state budget of the Czech Republic.

REFERENCES

[1] B. Subudhi and D. Jena, A differential evolution based neural network approach to nonlinear system identification, *Applied Soft Computing* 11(1) (2011) 861-871.

[2] B. Subudhi ,D. Jena, Nonlinear System Identification using Opposition Based Learning Differential Evolution and Neural Network Techniques, *Applied Soft Computing* 11(1) (2011) 861-871.

[3] A . M. Islam, S. Ghosh, S. Das, A. Abraham, S. Roy, A Modified Discrete Differential Evolution based TDMA scheduling scheme for many to one communications in wireless sensor networks, *Proceedings of IEEE Congress on Evolutionary Computation*, 2011,pp. 1950-1957.

[4] H. Bourlard , N. Morgan, Connectionist Speech Recognition: A Hybrid Approach, *The Kluwer International Series in Engineering and Computer Science*; v. 247, Boston: Kluwer Academic Publishers, 1994.

[5] H. Dhahri, A.M. Alimi, F. Karray: Opposition-based particle swarm optimization for the design of beta basis function neural network. *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Barcelona, Spain, 2010, pp.18-23.

[6] H. Dhahri, A.M. Alimi, F. Karray: Opposition-based differential evolution for beta basis function neural network. *Proceedings of IEEE Congress on Evolutionary Computation* , Barcelona, Spain, 2010, pp.1-8.

[7] A.M. Alimi, The Beta System: Toward a Change in Our Use of Neuro-Fuzzy Systems. *International Journal of Management,* Invited Paper, pp.15-19, 2000

[8] M.A Alimi., R. Hassine, and M. Selmi ," Beta Fuzzy Logic Systems: Approximation Properties in the SISO Case" , International Journal of Applied Mathematics & Computer Science, Special Issue on "Neuro-Fuzzy and Soft Computing" edited by D. Rutkowska and L.A. Zadeh, vol. 10, no. 4, (2000),pp. 857-875.

[9] C. Harpham , W. Dawson , R. Brown, A review of genetic algorithms applied to training radial basis function networks, *Neural Computing and Applications*, v.13 n.3, p.193-201, September 2004

[10] S.J. Ovaska, A. Kamiya, Y.Q. Chen, Fusion of soft computing and hard computing: computational structures and characteristic features. IEEE Trans. *Syst. Man Cybernet*. 36 (3). 439-448.

[11] P.M. Pawar, and R. Ganguli, Matrix crack detection in thin-walled composite beam using genetic fuzzy system. J. Intell. Mater. Syst. Struct. 16 (5). 395-409.

[12] X. Yao and Y. Liu (1998), Towards designing artificial neural networks by evolution, *Applied Mathematics and Computation*, 91(1): pp. 83-90.

[13] X. Yao and Y. Liu (1997), A new evolutionary system for evolving artificial neural networks, *IEEE Transactions on Neural Networks*, 8(3), pp. t694-713.

[14] X. Yao (1995), Designing Artificial Neural Networks Using Co-Evolution, *Proceedings of IEEE Singapore International Conference on Intelligent Control and Instrumentation*, pp.149-154.

[15] F. Mascioli and G. Martinelli (1995), a constructive algorithm for binary neural networks: The oil Spot Algorithm, *IEEE Transaction on Neural Networks*, 6(3), pp 794-797.

[16] A. Abraham, Optimization of Evolutionary Neural Networks Using Hybrid Learning Algorithms, IEEE International Joint Conference on Neural Networks (IJCNN02), 2002 IEEE World Congress on Computational Intelligence, Hawaii, ISBN 0780372786, IEEE Press, Volume 3, pp. 2797-2802, 2002.

[17] H. Dhahri, A.M. Alimi, F. Karray, The modified particle swarm optimization for the design of the Beta Basis Function neural networks, *Proc. Congress on Evolutionary Computation*, Hong Kong, China, 2008, pp. 3874-3880.

[18] C.F. Juang, C.M. Hsiao, and C.H. Hsu, Hierarchical Cluster-Based Multispecies Particle-Swarm optimization for Fuzzy-System Optimization. *IEEE Transactions on Fuzzy Systems*, **18**(1) 2010 ,14-26.

[19] X. Xu, Y. Li, Comparison between Particle Swarm Optimization, Differential Evolution and Multi-Parents Crossover, *Proceedings of international conference on computational intelligence and securit*y , 2007,pp. 124-127.

[20] R. storn, Differential evolution –a simple and efficient adaptive scheme for global optimization over continuous space. *J. global optim*. **4** (1995) 341-359.

[21] H. Dhahri, A. M. Alimi, "Automatic Selection for the Beta Basis Function Neural Networks", *Nature Inspired Cooperative Strategies for Optimization (NICSO),* pp. 461-474, 2007.

[22] S. Das, A. Abraham, and A. Konar, "Adaptive clustering using improved differential evolution algorithm," IEEE Transactions on Systems, Man and Cybernetics – Part A, IEEE Press, USA, vol. 38, issue 1, pp. 218-237, 2008.

[23] M. Pant , R. Thangaraj , C. Grosan , A. Abraham, Hybrid Differential Evolution – Particle Swarm Optimization Algorithm for Solving Global Optimization Problems 1, *Proceedings of third International Conference on Digital Information Management*, 2008, pp. 18-24

[24] D. Karaboga and B. Akay, A comparative study of Artificial Bee Colony algorithm, *Applied Mathematics and Computation*, 214(1),2009,108-132;

[25] H. Simon, Neural networks: a comprehensive foundation, New Jersey, Printice Hall, 1994.

[26] H. Dhahri, and A.M. Alimi, Hierarchical Learning Algorithm for the Beta Basis Function Neural Network, *Proc. Third International Conference on Systems, Signals & Devices*, Tunisia, 2005.

[27] M. Njah, A.M. Alimi, M. Chtourou and R. Tourki , Algorithm of Maximal Descent AMD for training Radial Basis Function Neural Networks , *Proc. IEEE International Conference on Systems, Man and Cybernetics: SMC'02*, Hammamet, Tunisia, October 2002.

[28] H. Dhahri, and M.A. Alimi, The Modified Differential Evolution and the RBF (MDE-RBF) Neural Network for Time Series Prediction", *Proc. International Joint Conference on Neural Networks: IJCNN'06*, Vancouver, July, pp 5245-5250, 2006.

[29] D, Karaboga, (2005). An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[30] D. E. Goldberg, (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

[31] Y. Chen, B. Yang, J. Dong, Time-series prediction using a local linear wavelet neural network, *Neurocomputing* 69 (2006) 449–465

[32] *R.M. Tong*, The evaluation of fuzzy models derived from experimental data, *Fuzzy Sets and Systems*, 4 (1980) 1-12.

[33] W. Pedrycz, An identification algorithm in fuzzy relational system. *Fuzzy Sets and Systems*, 13 (1984), pp. 153–167.

[34] C.W. Xu and Y.Z. Lu, Fuzzy model. Identification and self-learning for dynamic systems, *IEEE Trans. Syst. Man, Cyber*, (17) 4,683-689, 1987.

[35] N.K. Kasabov, J. Kim, M.J. Watts, and A.R. Gray, "FuNN/2 - A Fuzzy Neural Network Architecture for Adaptive Learning and Knowledge Acquisition", *presented at Inf. Sci*., 1997, pp.155-175.

[36] Kim, J., Kasabov, N.K.: HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems. *Neural Networks*(1999) 1301-1319

[37] Y. Chen, B. Yang, J. Dong, Nonlinear System Modeling via Optimal Design of Neural Trees, *Int. J. of Neural Systems*, vol.8, no.2,pp.125-137, 2004.

[38] H. Tamura, K. Tanno, H. Tanaka, C. Vairappan, Z. Tang, Recurrent Type ANFIS using Local Search Technique for Time Series Prediction, *in proc. IEEE Asia Pacific Conf. Circuits Sys.*, (2008), pp.380-383.

[39] S. Yilmaz, Y. Oysal. Fuzzy wavelet neural network models for prediction and identification of dynamical systems. *IEEE Transactions on Neural Networks*, 2010: 1599~1609

[40] E. Y., Cheu, C. Quek, , & Ng, S. K. (2008). TNFIS: Tree based neural fuzzy inference system. *In IEEE international joint conference on neuronal networks*, IJCNN, 2008, pp.398-405

[41] J.R. McDonnell, D. Waagen., Evolving recurrent perceptrons for time-series modeling, *IEEE Trans Neural Netw*. 1994;5(1):24-38.

[42] R. A. Aliev, B. G. Guirimov, R. R. Aliev, Evolutionary algorithm-based learning of fuzzy neural networks. Part 2: Recurrent fuzzy neural networks, *Fuzzy Sets and Systems*, 160 (17), September, 2009

[43] J. S. R. Jang, ANFIS: Adaptive network based fuzzy inference system, IEEE *Transactions on Systems, Man and. Cybernetics*, vol. 23, no. 3, pp. 665-685, 1993.

[44] A Hussain, A new neural network structure for temporal signal processing , *Proc Int Conf on Acoustics Speech and Signal Processing* (1997) ,pp. 3341-3344

[45] C. Xiang, W. Zhou, Z. Yuan, Y. Chen and Xi. Xiong, A new parameters joint optimization method of chaotic time series prediction, *International Journal of the Physical Sciences* Vol. 6(10), pp. 2565-2571, 18 May, 2011

[46] A. Abraham, Meta-Learning Evolutionary Artificial Neural Networks, Neurocomputing Journal, Elsevier Science, Netherlands, Vol. 56c, pp. 1-38, 2004.

[47] Y. Chen, B. Yang, J. Dong and A. Abraham, Time Series Forecasting Using Flexible Neural Tree Model, Information Sciences, Elsevier Science, Vol. 174, Issues 3/4, pp. 219-235, 2005.

[48] Y. Chen and A. Abraham, Tree-Structure based Hybrid Computational Intelligence: Theoretical Foundations and Applications, Intelligent Systems Reference Library Series, Springer Verlag, Germany, ISBN: 978-3-642-04738-1, 2009.