PSO-Based Update Memory for Improved Harmony Search Algorithm to the Evolution of FBBFNT’ parameters

S. Bouaziz, Adel M. Alimi, A. Abraham

Abstract—In this paper, a PSO-based update memory for Improved Harmony Search (PSOUM-IHS) algorithm is proposed to learn the parameters of Flexible Beta Basis Function Neural Tree (FBBFNT) model. These parameters are the Beta parameters of each flexible node and the connected weights of the network. Furthermore, the FBBFNT’s structure is generated and optimized by the Extended Genetic Programming (EGP) algorithm. The combination of the PSOUM-IHS and EGP in the same algorithm is so used to evolve the FBBFNT model. The performance of the proposed evolving neural network is evaluated for nonlinear systems of prediction and identification and then compared with those of related models.

Keywords—PSO-based update memory for Improved Harmony Search algorithm; Extended Genetic Programming; Flexible Beta Basis Function Neural Tree; nonlinear prediction systems, nonlinear identification systems.

I. INTRODUCTION

The Harmony Search (HS) algorithm proposed by Geem [1], was inspired from the jazz musical improvisation. The latter is done by a skilled musician (corresponding to the decision variable) who plays (generate) a note (value) to obtain a best harmony state (global optimum). The HS method is so considered as an evolutionary stochastic global optimization technique like genetic algorithm [38], bacterial foraging optimization algorithm [39, 40]. According to [1, 2], the HS algorithm outperforms the conventional mathematical optimization algorithms and the Genetic Algorithm [3]. Such algorithm has successfully been applied to solve some engineering applications such as robotics [4], routing problems [5], Transport energy modeling [6], etc.

On the other hand, Artificial Neural Network (ANN) is a growing interdisciplinary field which considers the systems as adaptive, distributed and mostly nonlinear, three of the elements found in the real applications. It is placed at the crossroads of various biological-inspired approaches where it is considered as an abstract simulation of a real nervous system. ANN is composed of an interconnected group of artificial neurons distributing in layers to model complex relationships between inputs and outputs of the studied problem. It mimics also the learning behavior of biological systems by updating its parameters (including interconnection weights and in certain cases transfer function parameters).

The know-how accumulated, so, through advanced work in the process of ANN creating and learning showed that its reliability can be conditioned by the appropriate structure; the connection ways between the nodes; the chosen transfer function; and the training algorithm. Many efforts have been provided in the literature to address these issues. Yao [7] is one of the first researchers who has exploited possible benefits arising from the interactions between ANNs and evolutionary computation, to design and evolve ANN, and in such case the model is noted Evolving Artificial Neural Network (EANN).

Recently several studies have proposed using HS algorithm for adjusting connected weights of artificial neural networks [8-11].

In this context is located the works presented in this paper. Indeed, a PSO-based update memory for Improved Harmony Search (PSOUM-IHS) algorithm is proposed to optimize the parameters of the Flexible Beta Basis Function Neural Tree (FBBFNT) model [12-15]. These parameters are the flexible Beta node parameters' and the weights encoded in the best structure found by the Extended Genetic Programming (EGP) [12, 13]. The new model is applied for nonlinear identification and prediction systems.

The paper is organized as follows: Section 2 describes the basic concepts of FBBFNT model. The PSOUM-IHS algorithm will be detailed in Section 3. A hybrid FBBFNT evolving algorithm, which combines EGP and PSOUM-IHS is the subject of Section 4. The set of some simulation results for nonlinear prediction and identification systems are provided in Section 5. Finally, some concluding remarks are presented in Section 6.

II. FLEXIBLE BETA BASIS FUNCTION NEURAL TREE MODEL

The initiative of using Beta function for designing Artificial Neural Network was introduced by Alimi [16]. This function has several advantages over the Gaussian function, such as its ability to generate more rich shapes (linearity, asymmetry, etc.) [17] and its great flexibility.

In this work, the Beta basis function neural network is encoded by the tree-based encoding method instead of the matrix-based encoding method [18], since this method is more flexible and gives amore modifiable and adjustable structure. The new model is called Flexible Beta Basis Function Neural Tree (FBBFNT). The FBBFNT is formed...
of a node set NS representing the union of function node set $F$ and terminal node set $T$:

$$NS = F \cup T = \{\beta_2, \beta_3, \ldots, \beta_N, f\} \cup \{x_1, \ldots, x_M\} \quad (1)$$

Where:
- $\beta_n$ ($n = 2, 3, \ldots, N$) denote non-terminal nodes and represent flexible Beta basis neurons with $n$ inputs and $N$ is the maximum degree of the tree.
- $f$ is the root node and represents a linear transfer function.
- $x_1, x_2, \ldots, x_M$ are terminal nodes and define the input vector values.

The output of a hidden function node is calculated as a flexible neuron model (Figure 1).

If a function node, i.e., $\beta_n$ is selected, $n$ real values are randomly generated to represent the weight connected between the selected node and its offspring. In addition, the Beta function has four adjustable parameters (the center $c_n$, the width $\sigma_n$ and the form parameters $p_n, q_n$) are randomly generated as flexible Beta operator parameters. For each function node, its total excitation is calculated by:

$$y_n = \sum_{j=1}^{n} w_j \ast x_j \quad (2)$$

Where $x_j$ ($j = 1, \ldots, n$) are the inputs of the selected node and $w_j$ ($j = 1, \ldots, n$) are the weights.

The output of node $\beta_n$ is then calculated by:

$$out_n = \beta_n(y_n, c_n, \sigma_n, p_n, q_n) =$$

$$\begin{cases} 
1 + \left(\frac{(p_n + q_n)(y_n - c_n)}{\sigma_n p_n}\right)^{p_n} \left[1 - \left(\frac{(p_n + q_n)(c_n - y_n)}{\sigma_n q_n}\right)^{q_n}\right] \\
\text{if } y_n \leq c_n - \frac{\sigma_n p_n}{p_n + q_n} c_n + \frac{\sigma_n q_n}{p_n + q_n} \\
0 \quad \text{else}
\end{cases} \quad (3)$$

The output layer yields a vector by linear combination of the node outputs of the last hidden layer to produce the final output.

A typical flexible Beta basis function neural tree model is shown in Figure 2. The overall output of flexible Beta basis function neural tree can be computed recursively by depth-first method from left to right.

### III. PSO-BASED UPDATE MEMORY FOR IMPROVED HARMONY SEARCH ALGORITHM: PSOUM-IHS

The Harmony Search (HS) algorithm searches the solution area as a whole to find the optimum element, which optimizes the fitness function. The steps in the procedure of harmony search are as follows [1]:

- **Step 1:** Problem formulation and parameter settings.
- **Step 2:** Initialize randomly the harmony memory.
- **Step 3:** Improvise a new harmony.
- **Step 4:** Update the harmony memory.
- **Step 5:** Checking stopping criterion

When the HS algorithm generates a new element, it considers all of the existing elements in the harmony memory (population) with fewer mathematical requirements. This characteristic makes the HS more flexible, the implementation easier and it is very versatile to combine HS with other meta-heuristic algorithms such as Particle Swarm Optimization (PSO) algorithm [19].

Moreover, in order to improve the adjusting characteristic of HS algorithm, Mahdavi et al. [20] suggested evolving the parameters instead of being fixed during the iterations. In fact, the authors suggested that $PAR$ (Pitch Adjustment Rate) increase linearly and $FW$ (width of the fret or bandwidth) decrease exponentially with iterations. Therefore, mathematic expressions were adapted into these parameters to follow the iteration change:

$$PAR = \frac{PAR_{max} - PAR_{min}}{\text{maxit}} \ast \text{currentIteration} + PAR_{min} \quad (4)$$

$$FW = FW_{max} \ast \exp(coef \ast \text{currentIteration}) \quad (5)$$

$$coef = \frac{\log(FW_{max}/\text{NI})}{\text{NI}} \quad (6)$$

Most of the decision variables in the new harmony are selected from the other elements stored in harmony memory. In addition, the new harmony vector may have the opportunity to take a place in the memory after his fitness test. Then, this vector might influence the convergence speed of the HS to the global optimum. We can note also that the harmony memory is stable in most of the time and does not provide a large variety of values to the
improvisation. Therefore, the HS has a low probability of generating a good-quality of the new harmony vector. For these reasons, we have to incorporate a mechanism to create a wide variety of values in memory while respecting their allowable ranges by a hybrid mechanism with PSO. This mechanism implicitly guides the global algorithm to converge to the optimal solution. The proposed algorithm (Figure 3) is called PSO-based update memory for Improved Harmony Search (PSOUM-IHS)[21]. In fact, the stochastic factors and the dynamic aspects of particle velocities can guide the system to the right areas of research in the workspace.

To ameliorate the performance of HS, we choose to apply the hybridization on the Improved Harmony Search algorithm instead of the basic version. Indeed, the memory vectors of IHS are considered as particles of the swarm and the new memory values for new improvisation as the new positions reached by these particles. The velocity parameter is calculated for each particle $i$ with the position $x_i$ according to the following equation:

$$v_i(t+1) = \Psi(t)v_i(t) + c_1 \varphi_1(p_i(t) - x_i(t)) + c_2 \varphi_2(p_g(t) - x_i(t))$$

Where $c_1$, $c_2$ (acceleration) and $\Psi$(inertia) are positive constant and $\varphi_1$ and $\varphi_2$ are randomly distributed number in $[0,1]$. In addition, $p_i$ corresponds to the best position of the current particles according to the best fitness and $p_g$ is the best position among all the particles obtained so far in the population. Each particle changes its position as the following equation:

$$x_i(t+1) = x_i(t) + (1 - \Psi(t))v_i(t+1)$$

IV. THE HYBRID ALGORITHM FOR EVOLVING FBBFNT MODEL

Evolving FBBFNT includes two issues which are architecture optimization and parameter adjustment. In this study, finding an optimal or a near optimal Beta basis function neural tree architecture is achieved by using Extended Genetic Programming (EGP) algorithm [12, 13].

The EGP which is an extended version of the standard genetic programming is formed by three mainly operators:

- Selection operator: is used to select two parent individuals from the population in order to procreate a new child by crossover/mutation operator.
- Crossover operator: is implemented by randomly taking selected two sub-trees in the individuals, and then swapping them.
- Mutation: four different mutation operators were used in the EGP to generate offspring from the parents. These mutation operators are as follows: changing one terminal node; changing all the terminal nodes; growing; pruning.

The parameters implanted in a FBBFNT are optimized by the PSO-based update memory for Improved Harmony Search (PSOUM-IHS) algorithm as described in section III.

So, to find an optimal or near-optimal FBBFNT model, structure and parameter optimization are used alternately. Combining of the EGP and PSOUM-IHS algorithms, a hybrid algorithm for evolving FBBFNT model is described as follows:
a) Randomly create an initial population (FBBFNT trees and its corresponding parameters);

\[ G = 0, \text{ where } G \text{ is the generation number of the evolving algorithm}; \]

\[ GlobalIter = 0, \text{ where } GlobalIter \text{ is the global iteration number of the learning algorithm}; \]

b) Structure optimization is achieved by the Extended Genetic Programming (EGP) with structure fitness function \( \text{Fit}_{\text{stru}}(i) = af_1(i) + \delta f_2(i) \) (9)

Where \( f_1 \) measures the RMSE between the target and output of the proposed model. The function \( f_2 \) measures the complexity of the FBBFNT model. \( a, \delta \) are user specified fitness coefficients that allow a trade-off between the objectives, \( a, \delta \in [0,1] \).

\[ f_1(i) = \frac{1}{P} \sum_{j=1}^{P} (y_j^i - y_{out}^i)^2 \] (10)

Where \( P \) is the number of samples, \( y_j^i \) and \( y_{out}^i \) are the desired output and the FBBFNT output of the \( j \)th sample.

\[ f_2(i) = \frac{\text{Size}_i - \text{Size}_{\text{Min} + 1}}{\text{Size}_{\text{Max}} - \text{Size}_{i + 1}} \] (11)

Where \( \text{Size}_i \) is the node number of the \( i \)th individual. The \( \text{Size}_{\text{Max}} \) and \( \text{Size}_{\text{Min}} \) are respectively the maximum and the minimum size of the tree.

c) If a better structure is found or a maximum number of EIP iterations is attained, then go to step (d),

\[ GlobalIter = GlobalIter + EGP\_Iter; \]

otherwise go to step (b);

d) Parameter optimization is achieved by the PSOUM-IHS algorithm. The architecture of FBBFNT model is fixed, and it is the best tree found by the structure search. The parameters (weights and flexible Beta function parameters) encoded in the best tree formulate a harmony matrix. The parameter fitness function is equal to the function \( f_1 \).

e) If the maximum number of PSOUM-IHS iterations is attained, or no better parameter vector is found for a fixed time then go to step (f);

\[ GlobalIter = GlobalIter + PSOUM-IHS\_Iter; \]

otherwise go to step (d);

f) If satisfactory solution is found or a maximum global iteration number is reached, then the algorithm is stopped; otherwise let \( G = G + 1 \) and go to step (b).

V. EXPERIMENTAL RESULTS

The FBBFNT model is applied to approximate the input/output map of nonlinear systems. Indeed, in order to prove the effectiveness of FBBFNT model, we compare its results with those provided by other learning methods in the literature. Before that the setting set of the FBBFNT is presented in the following Table 1. For all examples the illustrated results are obtained by averaging the results in 10 runs.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameter</th>
<th>Initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EGP</strong></td>
<td>Population size</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Crossover probability</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>Mutation probability</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>Maximum generation number</td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>Structure fitness settings</td>
<td>(0.95, 0.05)</td>
</tr>
<tr>
<td><strong>PSOUM-IHS</strong></td>
<td>Population size</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Maximum iteration number</td>
<td>4000</td>
</tr>
<tr>
<td></td>
<td>PARmin</td>
<td>0.00001</td>
</tr>
<tr>
<td></td>
<td>PARmax</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>HMCR</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>c1</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>c2</td>
<td>0.7</td>
</tr>
<tr>
<td><strong>Hybrid evolving algorithm</strong></td>
<td>Maximum global iteration number</td>
<td>40 000</td>
</tr>
<tr>
<td></td>
<td>Connected weights</td>
<td>rand[0, 1]</td>
</tr>
<tr>
<td></td>
<td>Beta center c</td>
<td>rand[min(x), max(x)]</td>
</tr>
<tr>
<td></td>
<td>Beta spread σ</td>
<td>rand[0, min(x) - max(x)]</td>
</tr>
<tr>
<td></td>
<td>Beta form parameters(p, q)</td>
<td>rand[0, 5]</td>
</tr>
</tbody>
</table>

A. Example 1: Mackey–Glass time series prediction

A time-series prediction problem can be constructed based on the Mackey–Glass [22] differential equation:

\[ \frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1 + x^2(t-\tau)} - bx(t) \] (12)

The settings of the experiment vary from one work to another. In our case, we take \( a = 0.2, b = 0.1, c = 10, \) and \( \tau = 17 \). These values are the same ones used by the comparison systems [12-15, 23-28]. As in the studies mentioned above, the task is to predict the value of the time series at point \( x(t) \), with using the inputs variables \( x(t), x(t - 6), x(t - 12) \), and \( x(t - 18) \) sample points are used in our study. The first 500 data pairs of the series are used as training data, while the remaining 500 are used to validate the model identified.

The used Beta basis function sets to create an optimal FBBFNT model with EGP&PSOUM-IHS system is \( \{\beta_2, \beta_3 / 3\} \cup \{x_1, x_2, x_3, x_4\} \), where \( x_i \ (i = 1, 2, 3, 4) \) denotes \( x(t), x(t - 6), x(t - 12), \) and \( x(t - 18) \), respectively.

After 86 global generations \( (G = 86) \), 1,675,425 global number of function evaluations, and 23,206 global iterations of the hybrid learning algorithm, an optimal FBBFNT model was obtained with RMSE 4.8855e-13. The RMSE value for validation data set is 4.8876e-13. The evolved FBBFNT_EGP&PSOUM-IHS architecture is as shown in figure 4. The evolved FBBFNT_EGP&PSOUM-IHS output and the desired output are shown in figure 5.

In addition, the memory footprint of the source code is of the order of 10,720 Kilo-bytes, knowing that the memory footprint of the input data is 40 Kilo-bytes. For the execution time is of the order of 2924 seconds.
The FBBFNT_EIP&PSOUM-IHS model is essentially compared with the FBONT [12], FBBFNT_EGP&OPSO [13], FBBFNT_EIP&OPSO [14], and FBBFNT_EIP&HBFOA [15] with the same initial setting’s values. The comparison is mainly based on the prediction error (RMSE) / Number of Function Evaluations (NFEs) compromise. In fact, for FBONT model RMSE is equal to 0.0076, NFEs is equal to 2,934,112 and for FBBFNT_EGP&OPSO, RMSE is equal to 0.0068, NFEs is equal to 1,966,825. Furthermore, for FBBFNT_EIP&OPSO model RMSE is equal to 0.0042, NFEs is equal to 1,616,648 and for FBBFNT_EIP&HBFOA, RMSE is equal to 1.8630e-09, NFEs = 6,004,148. It is clear that FBBFNT_EGP&PSOUM-IHS significantly reduces both the prediction error over the other four models and the number of function evaluations.

The FBBFNT_EIP&PSOUM-IHS network is also compared with Hierarchical multi-dimensional differential evolution for the design of Beta basis function neural network (HMDDE-BBFNN) [23] and the FNT model with Gaussian function as flexible neuron operator [24] and also with other systems. The HMDDE-BBFNN approach adopts for parameters: 50 to the population size, 10,000 to a total number of iterations, and 4 to the number of the hidden nodes. Moreover, the parameter settings of the FNT system are 30 to the population size, 135 as generation number, and 4 as hidden function unit number (with two hidden layers).

A comparison result of different methods for forecasting Mackey-Glass data is shown in Table 2. As observed, the FBBFNT_EIP&PSOUM-IHS achieves the lowest testing and training error.

### Table 2. Comparison of different methods for the prediction of Mackey-Glass time-series.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training error (RMSE)</th>
<th>Testing error (RMSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCMSPSO [26]</td>
<td>0.0095</td>
<td>0.0208</td>
</tr>
<tr>
<td>Fuzzy&amp;MRB [27]</td>
<td>0.000090</td>
<td>0.000884</td>
</tr>
<tr>
<td>LNF [28]</td>
<td>0.0199</td>
<td>0.0322</td>
</tr>
<tr>
<td>FNT [24]</td>
<td>0.0069</td>
<td>0.0071</td>
</tr>
<tr>
<td>HMDDE–BBFNN [23]</td>
<td>0.0094</td>
<td>0.0170</td>
</tr>
<tr>
<td>GA-BBFNN [25]</td>
<td>-</td>
<td>0.013</td>
</tr>
<tr>
<td>FBONT [12]</td>
<td>0.0074</td>
<td>0.0076</td>
</tr>
<tr>
<td>FBBFNT_EGP&amp;OPSO [13]</td>
<td>0.0061</td>
<td>0.0068</td>
</tr>
<tr>
<td>FBBFNT_EIP&amp;OPSO [14]</td>
<td>0.004194</td>
<td>0.004299</td>
</tr>
<tr>
<td>FBBFNT_EIP&amp;HBFOA [15]</td>
<td>5.3430e-10</td>
<td>1.8630e-09</td>
</tr>
<tr>
<td>FBBFNT_EGP&amp;PSOUM-IHS</td>
<td>4.8855e-13</td>
<td>4.8876e-13</td>
</tr>
</tbody>
</table>

### Example 2: Box and Jenkins’ Gas Furnace Problem

The gas furnace data of Box and Jenkins [29] was saved from a combustion process of a methane-air mixture. It is used as a benchmark example for testing prediction methods. The data set forms of 296 pairs of input-output measurements. The input \( u(t) \) is the gas flow into the furnace and the output \( y(t) \) is the \( \text{CO}_2 \) concentration in outlet gas. The inputs for constructing FBBFNT model are \( y(t - 1), u(t - 4) \), and the output is \( y(t) \). In this study, 200 data samples are used for training and the remaining data samples are used for testing the performance of the proposed model. The used instruction set is \( x = \{ \beta_2/3 \} \cup \{ x_1, x_2 \} \), where \( x_1 (i = 1, 2) \) denotes \( y(t - 1), u(t - 4) \), respectively.

After 17 global generations \( G = 27 \), 199,229 global number of function evaluations, and 4,564global iterations of the hybrid learning algorithm, an optimal FBBFNT model was obtained with RMSE 0.008135. The RMSE value for validation data set is 0.008773. In addition, the memory footprint of the source code is of the order of 2508 KB, knowing that the memory footprint of the input data is 7.008 KB. For the execution time is of the order of 1021 seconds.

The evolved FBBFNT_EGP&PSOUM-IHS architecture is as shown in Figure 6. Our method uses only three hidden function neurons for two hidden layers. The evolved FBBFNT_EGP&PSOUM-IHS output and the desired output are shown in Figure 7.
A comparison result of different methods for forecasting Jenkins-Box data is shown in Table 3.

**TABLE III. COMPARISON OF TESTING ERRORS OF BOX AND JENKINS.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Prediction error (RMSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANFIS model [30]</td>
<td>0.0845</td>
</tr>
<tr>
<td>FNN AP&amp;PSO [31]</td>
<td>0.0260</td>
</tr>
<tr>
<td>FuNN model [32]</td>
<td>0.0714</td>
</tr>
<tr>
<td>FNT [24]</td>
<td>0.0256</td>
</tr>
<tr>
<td>HMDDE-BBFNN[23]</td>
<td>0.2411</td>
</tr>
<tr>
<td>FBBFNT_EGP&amp;OPSO [13]</td>
<td>0.011618</td>
</tr>
<tr>
<td>FBBFNT_EIP&amp;OPSO [14]</td>
<td>0.00981</td>
</tr>
<tr>
<td>FBBFNT_EIP&amp;HBFOA [15]</td>
<td>0.009121</td>
</tr>
<tr>
<td>FBBFNT_EGP&amp;PSOUM-IHS</td>
<td>0.008773</td>
</tr>
</tbody>
</table>

It is clear from the results of Table 3, that the FBBFNT_EGP&PSOUM-IHS model gives the best prediction rate for the Jenkins-Box time-series.

**C. Example 3: Lorenz chaotic time series prediction**

The Lorenz system is a model of fluid motion between a hot surface and a cool surface [33]. It is described by the following nonlinear ordinary differential equations:

\[
\begin{align*}
\dot{x} &= \sigma(y - x) \\
\dot{y} &= -y - xz + rx \\
\dot{z} &= xy - bz
\end{align*}
\]  

(10)

In this example, the x-component in the Lorenz equations is used as the time series. The data that describe the Lorenz attractor, were generated by solving the system of differential equations, with \(\sigma = 10, r = 28\) and \(b = 8/3\). The data were used as inputs to the neural networks. The prediction is based on four past values \(x(t-4), x(t-3), x(t-2), \ldots, x(t-1)\) and thus the output pattern is \(nx = f(x(t-4), x(t-3), x(t-2), x(t-1))\).

From 1000 generated observations, the first 500 data pairs of the series are used as the training set and the last 500 are employed as test series. The used Beta basis function sets to create an optimal FBBFNT model with EGP&PSOUM-IHS system is \(NS = \{\beta_{2/3}\} \cup \{x_1, x_2, x_3, x_4\}\), where \(x_i (i = 1, 2, 3, 4)\) denotes \(x(t-4), x(t-3), x(t-2), x(t-1)\), respectively. After 14 global generations (\(G = 14\)), 227,904 global number of function evaluations, and 5115 global iterations of the hybrid learning algorithm, an optimal FBBFNT model was obtained with RMSE 2.5000e-11. The RMSE value for validation data set is 2.6092e-11. In addition, the memory footprint of the source code is of the order of 2480 KB, knowing that the memory footprint of the input data is 40 KB. For the execution time is of the order of 979 seconds. The evolved FBBFNT_EGP&PSOUM-IHS architecture is as shown in Figure 8. Our method uses only two hidden function neurons for one hidden layer. The evolved FBBFNT_EGP&PSOUM-IHS output and the desired output are shown in Figure 9.

Table 4 illustrates the comparison of the proposed algorithm with other models according to the training and testing errors. This table shows the performance of our approach for Lorenz chaotic time series comparing with the other model.

**TABLE IV. COMPARISON OF DIFFERENT MODELS OF LORENZ TIME SERIES PREDICTION.**

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE Training</th>
<th>RMSE Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>UD-SVM [34]</td>
<td>-</td>
<td>0.290</td>
</tr>
<tr>
<td>MLF-EKF [36]</td>
<td>0.0145</td>
<td>0.0408</td>
</tr>
<tr>
<td>MLP-BLM [36]</td>
<td>0.0174</td>
<td>0.0314</td>
</tr>
<tr>
<td>RNN-BPTT [36]</td>
<td>0.0227</td>
<td>0.0436</td>
</tr>
<tr>
<td>RNN-RTRL [36]</td>
<td>0.0229</td>
<td>0.0420</td>
</tr>
<tr>
<td>RNN-EKF [36]</td>
<td>0.0182</td>
<td>0.0352</td>
</tr>
<tr>
<td>RBLM-RNN [36]</td>
<td>0.0182</td>
<td>0.0304</td>
</tr>
<tr>
<td>LNF [28]</td>
<td>0.0039</td>
<td>0.0081</td>
</tr>
<tr>
<td>ABC_BBFNN [35]</td>
<td>-</td>
<td>0.076</td>
</tr>
<tr>
<td>FBBFNT_EGP&amp;PSOUM-IHS</td>
<td>2.5000e-11</td>
<td>2.6092e-11</td>
</tr>
</tbody>
</table>
D. Example 4: Nonlinear system identification

In this example, the nonlinear system [29] to be identified is expressed by:

\[ y_p(t + 1) = \frac{y_p(t)[y_p(t-1)+2][y_p(t)+2.5]}{8.5[y_p(t)]^2+[y_p(t-1)]^2} + u(t) \number{11} \]

Where \( y_p(t) \) is the output of the system at the \( t \)-th time step and \( u(t) \) is the plant input which is uniformly bounded in the region \([-2, 2]\). The identification model is as follows:

\[ y_p(t + 1) = f(y_p(t), y_p(t - 1)) + u(t) \number{12} \]

Where \( f(y_p(t), y_p(t-1)) \) is the nonlinear function of \( y_p(t) \) and \( y_p(t - 1) \) that will be input of our model and related works; and \( y_p(t + 1) \) will be the output from the neural models.

In this example, 500 data samples are used for training and 500 data samples are used for testing the performance of the evolved model. After the training is over, the identifier’s prediction ability has been tested for the input calculated as following:

\[ u(t) = \begin{cases} 2 \cos(2\pi t/100) & \text{if } t \leq 200 \\ 1.2 \sin(2\pi t/20) & \text{if } 200 < t \leq 500 \end{cases} \number{13} \]

The used Beta basis function sets to create an optimal FBBFNT model with EGP&PSOUM-IHS system is \( NS = \{ \beta_2/3 \} U \{ x_1, x_2 \} \), where \( x_i (i = 1, 2) \) denotes, \( y_p(t) \) and \( y_p(t-1) \), respectively.

After 10 global generations \((G = 10)\), 245,126 global number of function evaluations, and 5,469 global iterations of the hybrid learning algorithm, an optimal FBBFNT model was obtained with RMSE 1.0464e-12. The RMSE value for identification data set is 1.5441e-12. In addition, the memory footprint of the source code is of the order of 9092 KB, knowing that the memory footprint of the input data is 24 KB. For the execution time is of the order of 2533 seconds.

The evolved FBBFNT_EGP&PSOUM-IHS architecture is as shown in Figure 10. Our method uses only two hidden function neurons for one hidden layers. The evolved FBBFNT_EGP&PSOUM-IHS output and the desired output are shown in Figure 11. A comparison result of different methods for forecasting Lorenz data is shown in Table 5. Results show that applying the FBBFNT_EGP&PSOUM-IHS for the nonlinear plant identification improves the generalization error.

VI. CONCLUSIONS

In this paper, a new PSO-based update memory for Improved Harmony Search (PSOUM-IHS) algorithm is introduced to evolve the parameters of the Flexible Beta Basis Function Neural Tree (FBBFNT) model. These parameters include the parameters of Beta basis function nodes and connected weights. The PSOUM-IHS algorithm is combined with the Extended Genetic Programming for FBBFNT’s structure optimization. This combination can successfully optimize simultaneously the structure and the parameters of the FBBFNT. The results show that the FBBFNT_EGP&PSOUM-IHS method can effectively predict nonlinear systems such as Mackey-Glass chaotic time series, Lorenz–Box time series, Lorenz chaotic time series, and nonlinear plant identification.

ACKNOWLEDGMENTS

The authors would like to acknowledge the financial support of this work by grants from General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program. This work was also supported in the framework of the IT4 Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 by operational programme ‘Research and Development for Innovations’ funded by the Structural Funds of the European Union and state budget of the Czech Republic, EU.

REFERENCES
