2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Multi-agent Evolutionary Design of Flexible Beta Basis Function Neural Tree

M. Ammar, S. Bouaziz, Adel M. Alimi and Ajith Abraham

***Abstract*—Multi-Agent System (MAS) is a very active field that ensures global coherence between agents' interactions in a distributed way and implicit global control. Under the awareness of its power, the application of MAS was no more limited to very specific problems, but to almost application area: optimization, neural network, robotics, fuzzy system, etc. In the other side, a complex system of Artificial Neural Network called Flexible Beta Basis Function Neural Tree (FBBFNT) has reached a great level in the prediction search domain. In the purpose of enlarging the application of the algorithm to complex applications of the real problems, a new architecture of MAS was designed and applied to the FBBFNT process. This new multi-agent system based on communications and negotiations allowed the resolution of more complex prediction problems and the acceleration of the global convergence speed.**

I. Introduction

In nature life, human as an 'intelligent system' is a part of social system in which it operates and interacts with other humans distributed in the same environment. Humans in interactions can exchange information, negotiate decisions, share views, and resolve existing conflicts and they belong to different organizational structures [37]. The Multi-Agent System (MAS) considers a new philosophy of intelligent systems inspired from this nature view. The MAS is a sophisticated system comprising a set of interacting agents. It aims at providing solutions to inherently complex applications in distributed environment [15]—[17].

An agent is an intelligent autonomous entity such as a software program or a robot that is capable to reach its concerned objectives [18]. Originally, MAS considers a derivate branch from Distributed Artificial Intelligence (DAI) subfield of Artificial Intelligence (AI). Since its invention in 1970, DAI was presented a rich science and an evolving and interesting field of research [37]. It can be considered as an intersection with others domains such as artificial intelligence, management software engineering, organization, distributed systems, computer science, sociology, etc [30]. Moreover, it provides an efficient paradigm of distributing and coordinating a set of jobs, tasks and decisions between different agents to build coherent and interactive systems [17].

The extensive growth of the multi-agent field could be explained by the fact that many researchers have considered the Multi-Agent Systems as the current and future key for solving engineering problems [37]. It has the capacity to deal with heterogeneous, distributed, large and complex applications and environments in different area such as optimization, neural network [5], [40], robotics [27], fuzzy system [25], etc.

In this context, the Beta Basis Function Neural Network (BBFNN) was one of the complex applications. It showed a great performance in several success researches like classification [1], [3] (pattern recognition), prediction [12], [13], [31]—[35]. The network structure evolution and the parameter optimization are the two mainly issues that influence on the BBF neural network's performance. The BBFNN structure is not unique and depends on both, the treated problem and the method of conception. In addition, the parameters of BBFNN including connected weights and Beta parameters can be learned using back-propagation algorithm, genetic algorithm [7], particle swarm optimization algorithm [12] and so on. Furthermore, many attempts have been realized to evolve both structure and Beta parameters of the BBFNN such as Hierarchical Genetic Algorithm (HGA) [8] and Hierarchical Multi-dimensional Differential Evolution (HMDDE) [13].

Moreover, to have more flexible structure of the BBFNN, Bouaziz et al. [31]—[35] has used the tree-based encoding method for representing the BBF neural network. The new representation called Flexible Beta Basis Function Neural Tree (FBBFNT). This system adapted a simultaneous evolution of the structure and the parameters of the NN using different Evolutionary Computation algorithms such as Genetic programming [20], [31], Artificial Immune Systems [33], Particle Swarm Optimization, Differential Evolution, Bacterial Foraging Optimization Algorithm [39], Artificial Bee Colony [31], Harmony Search [26], and so on.

In this paper, a new architecture of multi-agent system was presented. It used a collection of three types of agent with different characteristics interconnected coherently. This system was designed for the FBBFNT model to accelerate the convergence and adapted it to more complex applications. It was called the Multi-Agent Evolving Flexible Beta Basis Function Neural Tree (MA_EFBBFNT)

The remaining paper is organized as follows: Section 2 presents the cumulative search to emerge the Flexible Beta Basis Function Neural Tree model. In section 3, the Multi Agent System is introduced with the different agents

M. Ammar, S. Bouaziz, Adel M. Alimi are with REsearch Group on Intelligent Machines (REGIM), University of Sfax, National School of Engineers (ENIS), BP 1173, Sfax 3038, Tunisia, (marwa.ammar.tn@ieee.org, souhir.bouaziz@ieee.org, adel.alimi@ieee.org)
Ajith Abraham is with [2]IT4Innovations, VSB-Technical University of Ostrava, Czech Republic and [1]Machine Intelligence Research Labs, WA, USA; (ajith.abraham@ieee.org)

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

implemented in our system. The new architecture of our multi-agent system which called MA_EFBBFNT is provided in Section 4. The set of some simulation results is the subject of Section 5. Finally, some concluding remarks are presented in Section 6.

II. Evolving FLEXIBLE Beta Basis Function Neural Tree

The Beta function was the transfer function used in the designing of the neural network. Alimi introduced this idea in 1997 [1]. It was adapted in our system because of its large flexibility and efficiency [2], [3], [28], [29], which exceeded the Gaussian function.

The Beta Basis Function Neural Network is a feed-forward neural network with three layers (input, hidden, output). In the hidden layer of the NN, the beta basis function was used as the non-linear transfer function. However, a linear transfer function was adopted in the output layer.

Furthermore, to extend the neural network structure for variable hidden layers, the classic matrix-based encoding was replaced by tree-based encoding. This new representation of the Beta basis function neural network was introduced by Bouaziz et al. in 2012 [31], [32], and the new model is called the Flexible Beta Basis Function Neural Tree (FBBFNT) (see Fig. 1).

The flexible nature of the tree has given the global system a comfortable flexibility for modifying and adjusting its structure. Therefore, the optimization of the FBBFNT had two parts; the tree structure evolution and the parameter evolution. A simultaneous evolution of architectures and learning parameters has been adapted using the Evolutionary Computation [20], [26], [31], [33], [39].

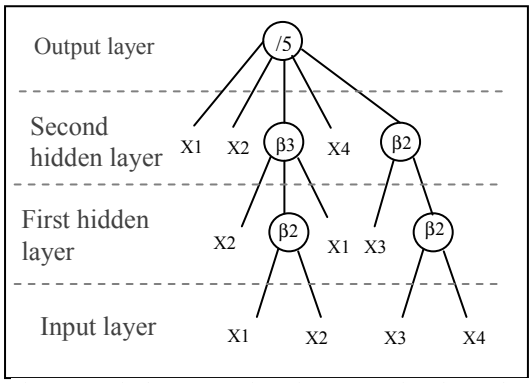

Fig. 1. A typical representation of FBBFNT: function node set F = {β2, β3, /5} and terminal node set T = {x1, x2, x3, x4}

A. Structure evolution

The structure optimization has taken a place after the initialization step. The initialization consisted to generate an initial population of trees with randomly structures and randomly parameters (node parameters and connecting weights).

In this work, the optimal structure or the near-optimal structure was achieved by using the Extended Genetic Programming (EGP) algorithm [31]. It was an extended version of The Genetic Programming (GP) paradigm introduced by Koza [20].

The EGP algorithm has three operators including selection, crossover and mutation, which are applied to each individual (or tree) of the population.

- **Selection operator:** is used to select two individuals from the actual population as a parent of the new child procreated by crossover/mutation operator.
- **Crossover operator:** is the swapping operation of two sub-trees from two different individuals randomly selected.
- **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 (replacing randomly a leaf node in hidden layer by a sub-tree); pruning (replacing randomly a beta operator node by a leaf node).

After that, each individual is evaluated according to the structure fitness function. The most adopted fitness function according to many search [6], [8], [13], [32] depends on the performance of the ANN on the training data (the Root Mean Squared Error (RMSE) between the target and output of the proposed model) and the complexity of the ANN related to the number of nodes and the number of layers.

B. Parameter evolution

After obtaining the best structure of the FBBFNT model using the Extended Genetic Programming, the Opposite-based Particle Swarm Optimization algorithm (OPSO) was applied to improve the performance of the parameters. These parameters concerned the Beta function parameters (centre: c, spread: s and the form parameters: p and q) [1], [2] of the corresponding node and the connecting weights.

The Opposition-based learning (OBL) proposed by Tizhoosh [14] was successfully applied to several problems. The opposite number $\bar{x}_i(t)$ is defined as follows:

For $x_i(t) \in [a_j, b_j]$

$$\bar{x}_i(t) = \begin{cases} \alpha_i (x_i(t) + \frac{a_j + b_j}{2}), if\ (x_i(t) \prec \frac{a_j + b_j}{2}) \\ \alpha_i (x_i(t) - \frac{a_j + b_j}{2}, if\ (x_i(t) \succ \frac{a_j + b_j}{2}) \end{cases} \quad (1)$$

Where $\alpha_i \in]0,1[$

The OPSO algorithm has the same basic concept as the PSO algorithm with adding the use of the opposite numbers to look for the solution in a limited search space. The idea was to divide the search space in two sub-spaces to accelerate the convergence rate and to opposite the numbers to bring them closer to the optimal solution.

The learning process of OPSO is described as follows:

- Step 0 (Initialization): The initial positions $x_i(t = 0)$ *(i = 1; :::;NP)* are generated randomly using this equation:

$$x_i(0) = a_j + rand_i(b_j - a_j) \quad (2)$$

Where $[a_j, b_j]$ is the search space of x_i

Then, the opposite population was computed according to the opposite based learning formula presented in equation 1. The initial velocities, $v_i(0)$ *with* $i = 1; :::;NP$, of all particles are randomly generated.

- Step 1 (Particle evaluation): Evaluate the performance of each particle in the population according to the beta neural system using the Root Mean Squared Error (RMSE) as a parameter fitness function:
- Step 2 (Velocity update): At iteration t, the velocity v_i of each particle i is updated using $pbest_i$ and $gbest_i$ according to this equation.

$$v_i(t+1) = \psi(t)v_i(t) + c_1\varphi_1(pbest_i(t) - x_i(t)) + c_2\varphi_2(gbest_i(t) - x_i(t)) \quad (3)$$

Where c_1, c_2 (acceleration), Ψ (inertia) are positive constant and φ_1 and φ_2 are randomly distributed number in [0, 1]. In addition, 'pbest' is the best fitness value achieved by the particle and the 'gbest' is the best fitness value obtained so far by any particle in the population. The velocity v_i is limited in $[-v_{max}, +v_{max}]$.

- Step 3 (Position update): Each particle updates its position and its opposite- position according depending on their velocities :

$$x_i(t+1) = x_i(t) + (1-\psi(t))v_i(t+1) \quad (4)$$

$$\bar{x}_i(t+1) = \bar{x}_i(t) + (1-\psi(t))v_i(t+1) \quad (5)$$

- Step 4 (*pbest* and *gbest* update): After updating the velocities and the positions of the whole population, the values of *pbest* (t) and *gbest*(t) are changed for the next iteration.
- Step 5 (End criterion): If the OPSO reached its objective or the maximum number of iterations prefixed, the process ends its execution.

So, to find the optimal or the near-optimal FBBFNT model, EGP (for structure optimization) and OPSO (for parameter optimization) are combined to evolve the whole system.
Different performance measures are used to evaluate the effectiveness of our model which are the RMSE training, RMSE testing and Evaluation Function Numbers (EFNs). The RMSE training and RMSE testing reflect the solution quality and the accuracy of the system. In addition, the EFNs compute the number of the evaluations function that reflects the convergence speed of the global algorithm. This number depends mainly on the problem treated and the neural network complexity (the number of input variables, tree degree fixed ...). Through the experimental results in the tables III and V, the EFNs have increased hugely when we tried to generalize the system and augmented the inputs number (from 4 to 19 or from 2 to 10) and the tree degree (from 3 to 6).
Therefore, this problem will be more acute when we try to deal with real problem data sets including a big number of inputs. Therefore, we have thought to modify the architecture of the system from a monolithic architecture to distributed architecture and to design a multi-agent system adapted to the ANN concept. This new system aims to reduce the convergence speed and the time complexity of the system.

III. THE MULTI-AGENT SYSTEM FOR THE FLEXIBLE BETA BASIS FUNCTION NEURAL TREE

A. The Multi-Agent System

The common goal of many researchers was to display an intelligent system that broke down the real-world problems. Since the mid-1990s, The Multi-Agent System (MAS) has attracted an international interest [37]. It has proved its efficiency in a host of different application domains. MAS has emerged as a new powerful paradigm of Artificial Intelligence (AI) which concretized the collective intelligence behaviors [18]. It focuses on the coordination of interactions between autonomous entities called agents in their common environment. Each agent was responsible to resolve set of tasks that beyond its individual capabilities.

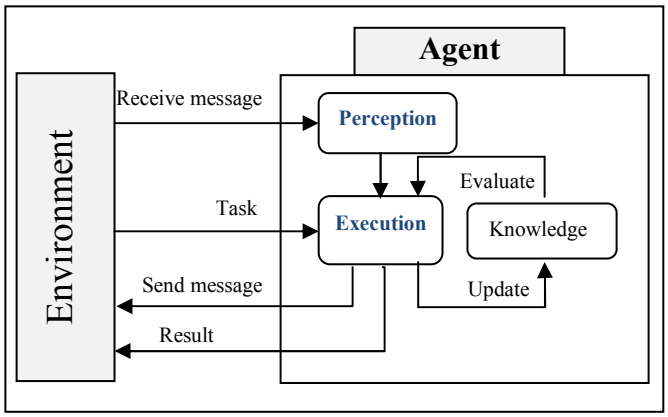

Fig. 2. Agent functional structure

An agent is an independent and autonomous entity that is able to interact, cooperate, coordinate and negotiate with other agents. The agents aim to solve the delegated tasks and ensure useful communications that allow the system to reach its global objective successfully. The figure 2 illustrates the agent design in our system and its general functional structure through the interaction with its environment. All the received messages are identified and interpreted with the *perception* functionality. Thus, the *execution* represents the resolution of the different tasks requested and the necessary knowledge update. Moreover, the agent can react in this system by sending a message to anther identified agent or by giving result. In our approach there are three types of agents with different functionalities:

1) Organizer agent:
It was an interactive agent. It has the responsibility of organizing the dynamic structure of the MAS including the starting step, the generation transition and the closing step. The organizer agent has the direct contact with the user; it sent the final solution for the appropriate problem defined by the user.

2) *Worker agent:* It was intelligent by optimizing some intern measures that can impact in its objectives. It has the more flexible (pro-active, reactive, social) behavior. It was pro-active agent by executing of the tasks and sending messages, reactive agent by answering requests and sending feedbacks, and has the social ability of negotiating with the other worker agents.

3) Principal agent: It was an active and interactive agent that was responsible for providing the admitted result.

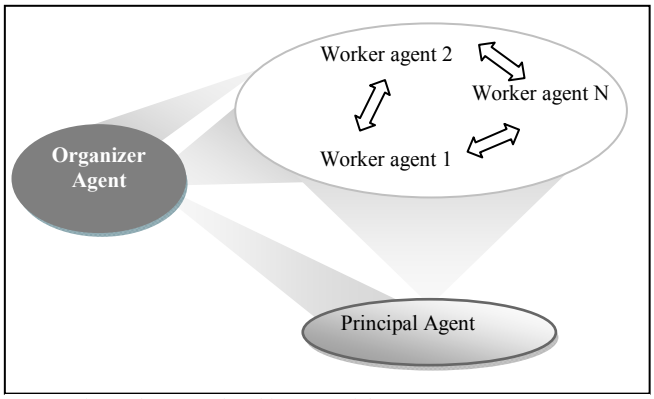

Fig. 3. the general architecture of the MA_FBBFNT system

B. The Multi-Agent Evolving Flexible Beta Based Function Neural Tree: MA_EFBBFNT

In this section, the multi-agent evolving FBBFNT which called MA_EFBBFNT was presented (see Fig. 3). As known, the multi-agent system was based on some protocols of communication between different agents to ensure the coherence. The communication protocol used is based on an asynchronous point-to-point communication.

Our algorithm was formed by these five general steps organized as follow:

Step0: Initialization (include the initialization of agents and the initialization of the EGP and OPSO algorithms)

Step1: The organizer agent started with dividing the population into N subpopulations and distributing them to the N worker agents.

Step2: The worker agents carried out the local FBBFNT algorithm.

Step2.1: When the worker agent reached an optimal tree structure (structure optimization by the EGP algorithm), it broadcasted it to the other worker agents.

Step2.2: A conflict of possible solutions appeared. Each worker agent opens a session of one-to-many negotiation. This negotiation relied on the complexity of the structure and the RMSE value and finished by adapting the best tree structure (with minimum complexity and minimum RMSE) for the rest of execution.

Step2.3: Another time, when the worker agent reached the set of optimal parameters (parameter optimization by the OPSO algorithm), it broadcasted it to the other worker agents.

Step2.4: Another conflict of possible solutions appeared. Each worker agent opens another session of one-to-many negotiation. This negotiation relied on the fitness (RMSE value) computed and finished by adapting the best parameters of the best tree for the next generation of population.

Step3: The final solutions of all worker agents were being sent to the principal agent forming its population. This agent executed the adapted FBBFNT algorithm and sent its 'Best_tree' to the organizer agent.

Step4: The organizer agent sent a message to the worker agents to look for the worst of them. After a negotiation, the worst worker agent was identified according to the worst solution have been sent to the principal agent before.

Step5: The organizer sent the 'Best_tree' to the worst worker to take a place in the next generation of population. The other worker agents update their population by their best local solutions.

IV. Experimental Results

The proposed MA_EFBBFNT system using the EGP algorithm for the structure optimization and the OPSO algorithm for the parameter optimization (see table I) is submitted to the Mackey-Glass time-series prediction and Box and Jenkins' Gas Furnace problem to evaluate its performance.

TABLE I
Optimization setting

Algorithm	Parameter	Initial value
Extended Genetic Programming (EGP)	Crossover probability	0.3
	Mutation probability	0.6
	Generation gap	0.9
Opposite based Particle Swarm Optimization (OPSO)	C_1	0.8
	C_2	0.8

A. Mackey-Glass time-series prediction

Several researchers have used the Mackey-Glass problem in order to compare the performances of its models. The Mackey-Glass problem proposed by Glass and Mackey [16] is a differential equation recognized as a benchmark problem.

$$\frac{dx(t)}{dx} = \frac{a * x(t-\tau)}{1 + x^{10}(t-\tau)} - b * x(t) \tag{6}$$

According to previous works, the input variables used are *x(t), x(t-6), x(t-12), x(t-18)* for predicting x(t +6). In our work, we predict the *x(t + 6)* value using the input variables *{x(t),x(t-1), x(t-2), x(t-3),.....,x(t-18)}*. It corresponds to 19 inputs to 1-output mapping.

The Mackey-Glass problem has generated 1000 observations; the first 500 pairs of data were considered as the training set and the last 500 were employed as test series [12], [21], [38].

B. Application to Mackey-Glass times-series prediction

According to the RMS Error and the global number of evaluation function NEFs, we have compared between the simulation results of MA_EFBBFNT for the Mackey-Glass times-series prediction and other methods from the literature. The problem was treated with two case of input variables number: 4 inputs (see table II) and 19 inputs (see table III).

From the experimental results presented in table II, the FBBFNT_EGP&OPSO provide better results more closely to the global optimum, but with more than 3 millions of function evaluations. In this context, the proposed multi-agent architecture for the algorithm (MA_EFBBFNT) gives better solution with minimum error and minimum NFEs. Moreover, according to table III, the experimental results approved that the MA_EFBBFNT overcomes the over dimensions imposed by the NN system and provides the best results with low error and a huge decrease in the number of evaluation function.

In this two cases treated, the integrating of the multi-agent architecture to the FBBFNT basic algorithm has ameliorated the performance of the system.

TABLE II
COMPARISON (RMSE TRAINING, RMSE TESTING AND NFES) BETWEEN MA_EFBBFNT AND DIFFERENT OTHER METHODS FOR MACKEY-GLASS PROBLEM WITH **4 INPUTS**

System	RMSE Training	RMSE Testing	NFEs
Classical RBF [22]	0.0096	0.0114	___
PSO-BBFN [12]	___	0.027	___
G_BBFNN [7]	___	0.013	___
CPSO [10]	0.0199	0.0322	___
FNT[39]	0.0071	0.0069	___
HCMSPSO [9]	0.0095	0.0208	___
FWNN-M [37]	0.00129	0.00114	___
HMDDE-BBFNN [13]	0.0094	0.0170	___
LNF [4]	0.00070	0.00079	
FBBFNT_EGP&PSO [33]	5.3000e-03	5.4000e-03	2,015, 358
FBBFNT_EGP&OPSO	5.5985e-04	5.8488e-04	1, 954,397
MA_EFBBFNT	**4.1262e-11**	**4.1310e-11**	**158,056**

TABLE III
COMPARISON (RMSE TRAINING, RMSE TESTING AND NFES) BETWEEN MA_EFBBFNT AND DIFFERENT OTHER METHODS FOR MACKEY-GLASS PROBLEM WITH **19 INPUTS**

System	RMSE Training	RMSE Testing	NFEs
FNT[39]	0,00276	0,00271	___
FBBFNT_EGP&OPSO	2.5444e-05	2.5132e-05	5,213,935
MA_EFBBFNT	**2.0622 e-06**	**2.0527 e-06**	**290,379**

C. Box and Jenkins' Gas Furnace problem

From the literature, another Benchmark problem using in the test of the prediction algorithms was the Box and Jenkins' Gas Furnace problem [11]. The process is based on the combustion of a mixture of methane-air. The output measurement is affected to Y(t) and the inputs measurement is a set constituted by $U(t-\tau_1)$ and $Y(t-\tau_2)$. The $U(t-\tau_1)$ presents the gas flow into the furnace at time $((t-\tau_1)$ with $\tau_1 \in \{1,2,3,4,5,6\})$ and the $Y(t-\tau_2)$ computes the concentration of the CO2 gas provided at time $((t-\tau_2)$ with $\tau_1 \in \{1,2,3,4,5\})$.

According to previous works (like works cited in [13], [32], [38], etc), the input-output variables used are {U(t-4),Y(t-1)}. The data set present 296 pairs of delayed input-output in different point of time.

For another simulation, 10 inputs variables {(U(t-6), U(t-5), U(t-4), U(t-3), U(t-2), U(t-1) and Y(t-1), Y(t-2), Y(t-3), Y(t-4)} are used in [39]. Thus, from the data set, 200 data samples are reserved for the training phase and the rest of data samples are used for the test of the system performance.

D. Application to Box and Jenkins' Gas Furnace problem

The same experimental process is applied for the Box and Jenkins' Gas Furnace problem using in the first case 2 inputs (see table IV) and in the second case 10 inputs (see table V). The following tables present the simulation result of the MA_EFBBFNT with other methods. The table V (first case) presents a comparison between our system in the two architectures (FBBFNT_EGP&OPSO and MA_EFBBFNT) and other methods from the literature. It is clear that the proposed algorithm in the proposed multi-agent architecture product results much better than the others. In addition, for the second case (10 inputs), the results presented in table V show the superior performance of the MA_EFBBFNT in terms of the performance measures used in this work (RMSE training, RMSE testing and NFEs).

So, our system has reached the best results with the two cases. It approved that the flexibility of the multi-agent architecture has a powerful affect to the performance of the system (low RMS Error) and especially to the time consuming (less NFEs).

This improvement of the system convergence speed makes this system more adaptable to deal with real data set problems.

TABLE IV
COMPARISON (RMSE TRAINING, RMSE TESTING AND NFES) BETWEEN MA_FBBFNT AND DIFFERENT OTHER METHODS FOR BOX AND JENKINS' GAS FURNACE PROBLEM WITH **2 INPUTS**

System	RMSE Training	RMSE Testing	NFEs
ANFIS model [19]	___	0.08544	___
FuNN model [21]	___	0.26720	___
HyFIS model [23]	___	0.25245	___
FNT[39]	0.000664	0.000701	___
FWNN-M [37]	0.01963	0.02324	___
HMDDE-BBFNN [13]	0.3745	0.2411	___
FBBFNT_EGP&PSO [33]	0.01735	0.01814	2,511,167
FBBFNT_EGP&OPSO	0.012454	0.01216	2,000,106
MA_EFBBFNT	**0.000041**	**0.000106**	**1,423,493**

TABLE V
COMPARISON (RMSE TRAINING, RMSE TESTING AND NFES) BETWEEN MA_FBBFNT AND DIFFERENT OTHER METHODS FOR BOX AND JENKINS' GAS FURNACE PROBLEM WITH **10 INPUTS**

System	RMSE Training	RMSE Testing	NFEs
FNT[39]	0.000291	0.000305	___
FBBFNT_EGP& OPSO	0.000044	0.000228	2,486,085
MA_FBBFNT	**4.0979e-05**	**1.8850e-04**	**185,576**

V. CONCLUSION

In this paper, a multi-agent system designed for artificial neural network which is called MA_EFBBFNT has been introduced. This system used a multi-agent architecture with three types of agents; worker agent, principal agent and organizer agent. It was implemented to a beta based function neural network using the tree representation. Moreover, two evolutionary algorithms were used for the optimization of the NN; the Extended Genetic Programming for the NN structure optimization and the Opposite based Particle Swarm Optimization for the NN parameters optimization. The system was evaluated using two prediction problems well known as benchmark problems. Its comparison with other methods from the literature has approved its superior performance, efficiency and speed.

ACKNOWLEDGMENT

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 program 'Research and Development for Innovations' funded by the Structural Funds of the European Union and state budget of the Czech Republic, EU.

REFERENCES

[1] Adel M. Alimi, "The Beta Fuzzy System: Approximation of Standard membership Functions", In Proc. 17eme Journées Tunisiennes d'Electrotechnique et d'Automatique, JTEA'97, pages 108–112, 1997.

[2] Adel M. Alimi, "What are the advantages of using the Beta neuro-fuzzy system? ", In Proc. IEEE/IMACS Multiconf. Computational Engineering in Systems Applications, CESAŠ98, volume 2, pages 339–344, 1998.

[3] Adel M. Alimi, "Beta Neuro-Fuzzy Systems", TASK Quarterly Journal, Special Issue on "Neural Networks", vol. 7, no. 1, pages 23–41, 2003.

[4] A. Miranian and M. Abdollahzade. Developing a Local Least-Squares Support Vector Machines-Based Neuro-Fuzzy Model for Nonlinear and Chaotic Time Series Prediction. IEEE Transactions on Neural Networks and Learning Systems, vol. 24, no. 2, pages 207–218, 2013.

[5] A. Quteishat *et al*. A neural network-based multi-agent classifier system, Neurocomputing 72 (2009) 1639–1647

[6] B. tak Zhang and H. Mühlenbein, "Balancing accuracy and parsimony in genetic programming". Evolutionary Computation, 1995.

[7] C. Aouiti, A.M. Alimi and A. Maalej. A Genetic Designed Beta Basis Function Neural Networks for approximating of multi-variables functions. In Proc. Int. Conf. Artificial Neural Nets and Genetic Algorithms, Springer Computer Science, pages 383–386, Prague, Czech Republic, 2001.

[8] C. Aouiti, Adel M. Alimi, F. Karray and A. Maalej, "The design of bate basis function neural network and beta fuzzy systems by a hierarchical genetic algorithm", Fuzzy Sets and Systems, vol. 154, no. 2, pages 251–274, 2005.

[9] 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, vol. 18, no. 1, pages 14–26, February 2010.

[10] F. V. D Bergh and A. P. Engelbrecht. A Cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pages 225–239, 2004.

[11] G.E.P. Box and G.M. Jenkins, "Time series analysis: forecasting and control". Holden-Day series in time series analysis and digital processing. Holden-Day, 1976.

[12] H. Dhahri, Adel M. Alimi and F. Karray, "Designing beta basis function neural network for optimization using particle swarm optimization", In IJCNN, pages 2564–2571, 2008.

[13] H. Dhahri, Adel M. Alimi and A. Abraham, "Hierarchical multidimensional differential evolution for the design of beta basis function neural network". Neurocomputing, vol. 97, pages 131–140, 2012.

[14] H. Tizhoosh, "Opposition-based Learning: A New Scheme for Machine Intelligence", Proceedings Int. Conf. Comput. Intell. Modeling Control and Autom, Vol. I, pp. 695-701, 2005.

[15] I. Dzitac, B.E. B˘arbat, "Artificial Intelligence + Distributed Systems = Agents". Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844, Vol. IV ,No. 1, pp. 17-26, 2009.

[16] J. Ferber, Les systèmes multi-agents: Vers une intelligence collective. InterEditions, Paris, 1995.

[17] J. Ferber, Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence, Addison Wesley, London, 1999.

[18] J. Ferber, O. Gutknecht1, and M. Fabien, "From Agents to Organizations: an Organizational View of Multi-Agent Systems". Agent-Oriented Software Engineering (AOSE) IV, P. Giorgini, Jörg Müller, James Odell, eds, Melbourne, July 2003, LNCS 2935, pp. 214-230, 2004

[19] J. Nie. Constructing fuzzy model by self-organizing counterpropagation network. IEEE Transactions on

Systems, Man, and Cybernetics, vol. 25, no. 6, pages 963–970, 1995.

[20] J.R. Koza, "Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems". Rapport technique, Stanford, CA, USA, 1990.

[21] J-S. R. Jang and C-T. Sun. Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[22] K.B Cho and B.H Wang. Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction. Fuzzy Sets and Systems, vol. 83, no. 3, pages 325–339, November 1996.

[23] K. K. Nikola, K. Jaesoo, J. W. Michael and R. G. Andrew. FuNN/2 – A Fuzzy Neural Network Architecture for Adaptive Learning and Knowledge Acquisition. Information Sciences, vol. 101, no. 3-4, pages 155–175, 1997.

[24] L. Glass and M. C. Mackey, "Pathological physiological conditions resulting from instabilities in physiological control systems". Ann. NY. Acad. Sci, vol. 316, pages 214–235, 1979.

[25] L. Mengual, J. Bobadilla, G. Triviño. "A fuzzy multi-agent system for secure remote control of a mobile guard robot", (2004) *Lecture Notes in Artificial Intelligence* (Subseries of Lecture Notes in Computer Science), 3034, pp. 44-53.

[26] M. Ammar, S. Bouaziz, Adel M. Alimi, "Hybrid Harmony Search algorithm for Global Optimization". the Fifth World Congress on Nature and Biologically Inspired Computing (NABIC) , pp. 69–75, Fargo-USA, 12-14 August 2013.

[27] M.C.G. Quintero , J.A.O. Lopez, R.F.A. Bertel, "Coordination mechanisms for a multi-agent robotic system applied to search and target location". IX Latin American Robotics Symposium and IEEE Colombian Conference on Automatic Control, 2011 IEEE, Oct. 2011, pp.1-6

[28] M. Masmoud, M. Same and Adel M. Alimi, "Beta Neuro-Fuzzy Systems". International Journal of Electronics, vol. 87, no. 6, pages 675–682, 2000

[29] M. Njah, Adel M. Alimi and M. Chtourou, "A learning algorithm for the Beta neuro-fuzzy network". In Proc. Int. Conf. Artificial and Computational Intelligence for Decision, Control and Automation, pages 76–81, Monastir, Tunisia, 2000.

[30] M. Wooldridge, An Introduction To Multi Agent Systems. West Sussex, Willey, 2002.

[31] S. Bouaziz, H. Dhahri, Adel M. Alimi and A. Abraham, "Evolving Flexible Beta Basis Function Neural Tree Using Extended Genetic Programming & Hybrid Artificial Bee Colony", submitted in Applied Soft Computing, December 2012.

[32] S. Bouaziz, H. Dhahri and Adel M. Alimi, "Evolving flexible beta operator neural trees (FBONT) for time series forecasting", In Proceedings of the 19th international conference on Neural Information Processing - Volume PartIII, ICONIP'12, pages 17–24, Berlin, Heidelberg, 2012. Springer-Verlag.

[33] S. Bouaziz, Adel M. Alimi and A. Abraham, "Extended Immune Programming and Opposite-based PSO for Evolving Flexible Beta Basis Function Neural Tree", IEEE International Conference on Cybernetics, pp. 13 –18, Lausanne Switzerland, 13-15 June 2013.

[34] S. Bouaziz, H. Dhahri and Adel M. Alimi, "Evolving Flexible Beta Basis Function Neural Tree for nonlinear systems", International Joint Conference on Neural Networks, Dallas Texas, 4-9 August 2013.

[35] S. Bouaziz, H. Dhahri, Adel M. Alimi and A. Abraham, "A Hybrid Learning Algorithm For Evolving Flexible Beta Basis Function Neural Tree Model", Neurocomputing, vol. 117, pp. 107–117, 2013.

[36] S. Yilmaz and Y. Oysal. Fuzzy wavelet neural network models for prediction and identification of dynamical systems. IEEE Transactions on Neural Networks,vol. 21, no. 10, pages 1599–1609, October 2010.

[37] W. Gerhard, "Multiagent systems: a modern approach to distributed artificial intelligence", 1999, The MIT Press ISBN 0-262-23203-0

[38] Y. Chen, B. Yang, J. Dong and A. Abraham. "Time-series forecasting using flexible neural tree model", Inf. Sci., vol. 174, no. 3-4, pages 219–235, August 2005.

[39] Y. Jarraya, S. Bouaziz, Adel M. Alimi, A. Abraham, "A Hybrid Computational Chemotaxis in Bacterial Foraging Optimization Algorithm for Global Numerical Optimization", IEEE International Conference on Cybernetics, Lausanne Switzerland, pp. 213 –218, 2013.

[40] Yong S. Choi, Suk I. Yoo and J. Lee. "Neural Network Based Multi-agent Information Retrieval System". Third International Symposium, IDA-99 Amsterdam, The Netherlands, August 9–11, 1999 Proceedings Volume 1642, 1999, pp 499-511.