A Hybrid Algorithm of Electromagnetism-like and Genetic for Recurrent Neural Fuzzy Controller Design

Ching-Hung Lee, *Member, IAENG* , Che-Ting Kuo, Hao-Han Chang, Jen-Chieh Chien, and Fu-Kai Chang

Abstract—**Based on the electromagnetism-like algorithm (EM), we propose a novel hybrid learning algorithms which is the improved EM algorithm with genetic algorithm technique (IEMGA) for recurrent fuzzy neural system design. IEMGA are composed of initialization, local search, total force calculation, movement, and evaluation. They are hybridization of EM and GA. EM algorithm is a population-based meta-heuristic algorithm originated from the electromagnetism theory. For recurrent fuzzy neural system design, IEMGA simulates the "attraction" and "repulsion" of charged particles by considering each neural system parameters as an electrical charge. The modification from EM algorithm is the neighborhood randomly local search is replaced by GA and the competitive concept is adopted for IEMGA. For gradient information free system, IEMGA is proposed to treat the optimization problem. Besides, IEMGA consists of EM and GA to reduce the computation complexity of EM. IEMGA is used to develop the update laws of RFNN for nonlinear system control problem. Finally, several illustration examples are presented to show the performance and effectiveness of IEMGA.**

Index Terms—**Electromagnetism-like algorithm, genetic algorithm, fuzzy neural system, nonlinear control**

I. INTRODUCTION

There are many literatures address in the training and designing of neural fuzzy systems [5, 10-12, 14-16, 19-20, 22-23, 26]. To solve this problem, back-propagation (BP) algorithm is widely used and is a powerful training technique [12, 16-19, 22]. This may obtain a local minimum rapidly and cannot find the global solution. Hence, training the neural network and finding the global optimization are important. Recently, several algorithms are proposed by the observation of real-world systems, such as, genetic algorithm (GA), evolutionary algorithm (EA), particle swarm optimization (PSO), immune algorithm (IA), etc [6-8, 10-11, 14-15, 27-28, 30, 32, 33]. Recently, a novel meta-heuristic algorithm: electromagnetism-like (EM) mechanism, for global optimization was proposed [1-4, 31, 34]. EM algorithm is originated from the electromagnetism theory in physics which simulated the electromagnetism theory of physics by considering each particle to be an electrical charge. Subsequently, the movement of attraction and repulsion is introduced by Coulomb's law. It has advantages of multiple search, global optimization, and faster convergence procedure and simultaneously evaluates many point in search space, they are more likely to find the better solution [1-4, 31, 34]. However, the local search procedure of EM is stochastic. Hence, the major drawback of EM is highly computation complexity. In order to improve performance of EM, a modified local search phase and the competition concept are adopted.

In recent years, fuzzy systems and neural networks are being used successfully in many application areas [5, 7, 10, 12, 14-16, 19-20, 24-26, 28-29, 33-34]. Based on the approximation ability, many adaptive control techniques are accompanied with them for approximation of system functions or controllers. A major drawback of the existing feed-forward neural fuzzy systems is that their application domain is limited to static problems due to the network structure. In literature [20], a recurrent fuzzy neural network (RFNN) system is proposed to identify and control nonlinear systems. It is more suitable than feed-forward neural network for temporal problems. By the way, some other recurrent fuzzy neural systems have been proposed [9, 13-14, 20, 21-23, 27, 37]. They have the ability of storing system past information. With the advantages, this study develops a recurrent fuzzy neural network-based control scheme for nonlinear systems.

This paper proposes the improved EM algorithm with genetic algorithm technique (IEMGA). The major modification from EM algorithm is the randomly neighborhood local search which is replaced by GA, and the competitive selection concept is adopted for reducing the computation complexity. The IEMGA has the ability of multiple searches, global optimization, and less computation complexity. Furthermore, IEMGA does not need any gradient information for optimization process. As a result of these advantages, we use IEMGA to solve nonlinear system control problem.

The paper is organized as follows. Section II introduces the electromagnetism-like algorithm for recurrent fuzzy neural system. In Section III, the hybrid algorithm IEMGA for RFNN controller design is introduced. Section IV shows the simulation and comparison results of nonlinear system control is shown to demonstrate the performance of the proposed IEMGA. Finally, conclusion is given.

II. ELECTROMAGNETISM-LIKE ALGORITHM FOR RECURRENT FUZZY NEURO SYSTEM

A. Recurrent Fuzzy Neural Network (RFNN)

Many results have been obtained by using fuzzy neural networks (FNNs) approach for system identification and control [5, 12, 15, 26]. However, there are some disadvantages of the FNNs such as their application domains

This work was supported in part by the National Science Council, Taiwan, R.O.C., under contracts NSC-95-2221-E-155-068-MY2.

Ching-Hung Lee is with Department of Electrical Engineering, Yuan-Ze University, Chung-li, Taoyuan 320, Taiwan. (phone: +886-3-4638800, ext: 7119; fax: +886-3-4639355, e-mail: chlee@saturn.yzu.edu.tw).

are limited in static problems because of feedforward structure. It is inefficient when the FNN is applied in the temporal problems. Hence, the recurrent fuzzy neural network (RFNN) is proposed [20], which is more suitable for solving temporal problems and describing dynamic systems than the FNN. It is more effective and more adaptive than the conventional FNN with non-adaptive fuzzy reasoning [9, 13-14, 17, 19-20, 27, 33].

The schematic diagram of RFNN is shown in Fig. 1, where G denotes the Gaussian membership function. The RFNN system is inherently a recurrent multilayered connectionist network for realizing fuzzy inference using dynamic fuzzy rules. In layer 2, the feedback networks are existed to afford the dynamic properties. Each dynamic fuzzy if-then rule in RFNN consists of n external inputs $x_1, x_2, \ldots, x_n$, and output y, which is in the form of

Rule j: IF z_1 is A_{1j} and … z_n is A_{nj}, THEN y is ω_j (1)

where input linguistic variable z_j is $O_{ij}^{(2)}(k-1)\cdot\theta_{ij}+O_i^{(1)}(k)$ which includes the current input and the past information, A_{1j} is a fuzzy set represented by Gaussian function and ω_j is the consequent part parameter for inference output y. In the following, we indicate the signal propagation and the basic function of each layer, $x_i^{(l)}$ and $O_i^{(l)}$ denote the node input and output; the superscript (l) denotes the lth layer and the subscript i denote the ith input.

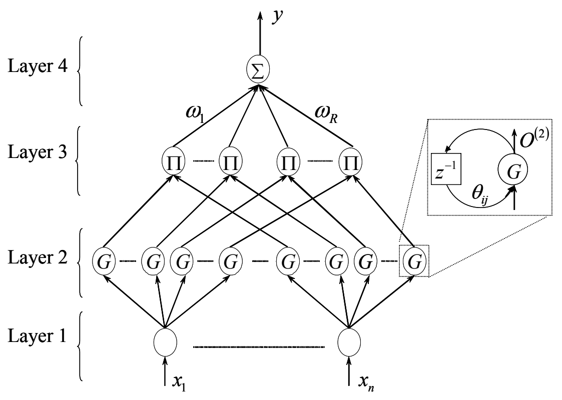

Figure 1: Network diagram of RFNN system [20].

Layer 1: Input Layer

Layer 1 accepts input variables and its nodes represent input variables. The corresponding output is

$$O_i^{(l)}(k)=x_i(k) \tag{2}$$

where $x_i(k)$, i=1, 2, …, n, represent the input variables.

Layer 2: Membership Layer

Layer 2 is used to calculate Gaussian membership grade, i.e.,

$$O_{ij}^{(2)}(k)=\exp\left[-\frac{\left(z_j(k)-m_{ij}\right)^2}{\left(\sigma_{ij}\right)^2}\right]. \tag{3}$$

As above, z_j is the fuzzy input linguistic variable, z_j= $O_{ij}^{(2)}(k-1)\cdot\theta_{ij}+O_i^{(1)}(k)$. In addition, here m_{ij} and σ_{ij} are the center and the width of the Gaussian function, θ_{ij} is the adjustable parameter of feedback layer. Obviously, the RFNN can store the past information [20]. And if we set θ_{ij}=0, the RFNN can be reduced to an FNN system [5, 20].

Layer 3: Rule Layer

Nodes in layer 3 represent fuzzy rules. Links before layer 3 represent the preconditions of the rules, and the links after layer 3 represent the consequences of the rule nodes. The product operation is used here i.e.,

$$O_j^{(3)}(k)=\prod_i O_{ij}^{(2)}(k). \tag{4}$$

Layer 4: Output Layer

Layer 4 is the output layer which is used to implement the defuzzification operation. Each node is for actual output to be pumped out this system. The links between layer 3 and layer 4 are connected by the weighting value ω_j, i.e.,

$$y=O^{(4)}(k)=\sum_{j=1}^{R}\omega_j O_j^{(3)}(k) \tag{5}$$

where ω= [ω_1, ω_2, …, ω_R]^T is the weighting vector. As above description, it has adjustable parameters m, σ, θ, and ω, which is denoted by $W=[m,\sigma,\theta,\omega]^T$.

B. Electromagnetism-like (EM) Algorithm

This section introduces the electromagnetism-like algorithm (EM) for training RFNN systems. EM algorithm was developed to simulate the electromagnetism theory by each sample point to be a particle [1]. The EM for optimization problems with lower and upper bound is in the form of

Minimize $f(x)$

Subject to $x\in S$, (6)

where $S=\{x\in\Re^n \mid l_k\le x_k\le u_k, l_k, u_k\in\Re, k=1,\ldots,n\}$ and the parameters are defined as: n : dimension of the problem, u_k : corresponding upper bound, l_k : corresponding lower bound, $f(x)$: pointer to the function that is minimized.

Herein, each particle x represents a solution and a particle is associated with each particle which is depended on the fitness function $f(x)$. EM utilizes the mechanisms of attraction and repulsion to put the points towards to the optimum. By the Coulomb's law, the magnitude of force is proportional to product of the particles and inversely proportional the distance between two particles. There are three phases in the learning algorithm EM which are "Initialization," "Evaluation," and "Operation."

M	**…**	σ	**…**	θ	**…**	ω	**…**

Figure 2: Particle representation of RFNN.

B.1 Initialization Phase

The real-value coding technique is used to represent a solution to a given problem. Each particle is encoded as a vector of real numbers with the same lengths as the solution vector. Each particles denotes a weighting vector $W=[m,\sigma,\theta,\omega]^T$ shown in Fig. 2 and the EM is utilized to find the optimal value $W^*=[m^*,\sigma^*,\theta^*,\omega^*]^T$.

Typically, initial particles are randomly chosen from a feasible solution region. "Initialization phase" is used to generate m initial particles. At first, the feasible region of solution for RFNN parameters should be defined. In addition, the training cycle is chosen to be the termination condition.

B.2 Evaluation Phase

This phase is used to calculate the fitness values of entire particles. Each particle is evaluated by the given fitness function to decide its survival or extinction in the next generation. Evaluation phase helps us to find superiority particles by determining the value of RMSE (or fitness value).

A particle with better fitness value has the higher possibility to survive. Three steps should be done in this phase- fitness values evaluation, fitness ranking, and best particle definition. To evaluate the performance of each particle in training RFNN system, we define the root-mean-square-error (RMSE) as

$$f(x) \equiv \sqrt{\sum_{k=1}^{N} e^2(k) / N} \tag{7}$$

where e denotes the approximated error and N denotes the data number. The corresponding fitness function is

$$g(x) = \frac{1}{1 + \text{RMSE}}. \tag{8}$$

Subsequently, all particles are ranked and indexed by the corresponding of fitness value. Finally, the particle having the largest fitness value (minimum RMSE) is stored in x^{best}.

B.3 Operation Phase

As description of literature [1-4, 31, 34], there are three steps in the EM operation phase. They are "local search", "total force calculation", and "movement", respectively.

B.3.1 Local Search of EM Algorithm

Local search step is used to gather the local information for each particle x^j. A new particle y is moved along the direction of x^j with the maximum feasible random step length δ ($\max_k\{u_k - l_k\}$). A simple neighborhood search is utilized which works on a small ball or radius δ around the particle. If the new particle y has better fitness value, the particle x^j is replaced by y. Then a new best particle is re-defined in this step. Details of local search can be found in literature [1, 17, 19].

By the results of [17, 18], EM algorithm has the properties of rapid convergence and global optimization. However, it spends a highly complexity computation for each generation.

B.3.2 Total Force Calculation

In this step, a particle is assigned to each particle of the population like electromagnetic charges. The charge q^i of particle x^i is determined by

$$q^i = \exp\left[-n \times \frac{f(x^i) - f(x^{best})}{\sum_{k=1}^{m}[f(x^k) - f(x^{best})]} \right], \quad i = 1,2,...m \tag{9}$$

the force is inversely propositional to the distance between two particles and directly proportional to the product of their chargers. Hence, the total force vector exerted on x^i computed by the superposition principle as follows

$$F^i = \begin{cases} \sum_{j \neq i}^{m} (x^j - x^i) \cdot \frac{q^i q^j}{\|x^j - x^i\|^2} & \text{if} \quad f(x^j) < f(x^i) \\ \sum_{j \neq i}^{m} -(x^j - x^i) \cdot \frac{q^i q^j}{\|x^j - x^i\|^2} & \text{if} \quad f(x^j) \geq f(x^i). \end{cases} \tag{10}$$

After comparing the RMSE values, (i.e.,$f(x)$), the direction of the forces between the particle and the others is selected. For two particles, the one has a better RMSE value attracts the other one. On the other hand, the particle with larger RMSE repels the others.

B.3.3 Movement

After determining the total force vector F^i, particle x^i moves in the direction of the total force by a random step length, i.e.,

$$x^i = x^i + \lambda \frac{F^i}{\|F^i\|}(RNG) \quad i = 1, 2, ..., m, \tag{11}$$

$$RNG = \begin{cases} u_k - x_k^i & \text{if} \quad F_k^i > 0 \\ x_k^i - l_k & \text{if} \quad F_k^i \leq 0 \end{cases} \quad k = 1, 2, ..., n \tag{12}$$

where the random step length λ = random (0, 1), and RNG is a vector whose components denotes the allowed feasible movement toward the upper bound, u_k, or lower bound l_k.

III. Hybrid Algorithm IEMGA for RFNN System

We introduce an improved EM algorithm with genetic algorithm (GA) technique, we called it IEMGA. Beacuse GA do not require derivative information, the most appropriate applications are problems where gradient information is unavailable or difficult to obtain. In real-world applications, precise system sensitivity is usually unknown. In addition, due to its global optimization capability, GAs has become another useful tool to the automatic adjustment of network parameters. For this reason, there has been a growing interest in GAs for FNN design [7-8, 14, 29, 32]. The IEMGA method contains the reproduction (competitive selection), crossover, and mutation. IEMGA does not need any gradient information and it is capable of decreasing the computational complexity. As a result of these advantages, we use IEMGA to solve nonlinear system control and the optimization of PID controller design.

A. Improved EM Algorithm Using GA Technique (IEMGA)

IEMGA combines the advantages of EM and GA algorithms to result high speed convergence and less computation complexity. Moreover, it does not need any system gradient information. The major modification from EM algorithm is the randomly neighborhood local search is replaced by GA. Figure 3 depicts the flow chart of the IEMGA algorithm.

Figure 4 shows the flow chart of the modified local search. After evaluation phase, the elites concept is adopted to implement the competitive selection. The 50% front particles are selected to enter the local search. The other 50% particles are discarded. Subsequently, the GAs optimization procedures including reproduction, crossover, and mutation are used to generate the better particles. Finally, the new particles and the remained 50% particles are combined to be the new population.. r is a floating-point number randomly between 0 and 1 for each chromosome. P_c and P_m are crossover rate and mutation rate, respectively.

A.1 Genetic Algorithm

Population size M is the number of solutions to parallel search in each generation. Large population size is difficult to fall into the area of best solutions. Therefore, for the setting of this parameter, a balance must be achieved between solution quality and operation time. In this paper, M is equal to the half of total particles according the competitive selection, i.e., $M = m/2$.

A.1.1 Reproduction by Competitive Selection

Fitness function is used to judge whether the chromosome is good or not. Chromosomes with lower fitness values may be easily eliminated during reproduction, highlighting the feature of "the fittest survival in natural selection" in the biosphere.

After fitness function calculation, population will decide whether to keep or get rid of the chromosomes. The species with high fitness is more ascendant than those with low fitness in the reproduction. The reproduced chromosomes will be placed in the mating pool, waiting for the next operation procedure. During the process of reproduction operation is usually divided into two ways [7, 8, 14]:

(1) Competitive selection
(2) Roulette wheel selection.

In this study, we choose the first method. Competitive selection is to randomly pick up more than two chromosomes to compare their fitness values. The one with a higher fitness value will be reproduced while the one with a lower fitness value will be eliminated.

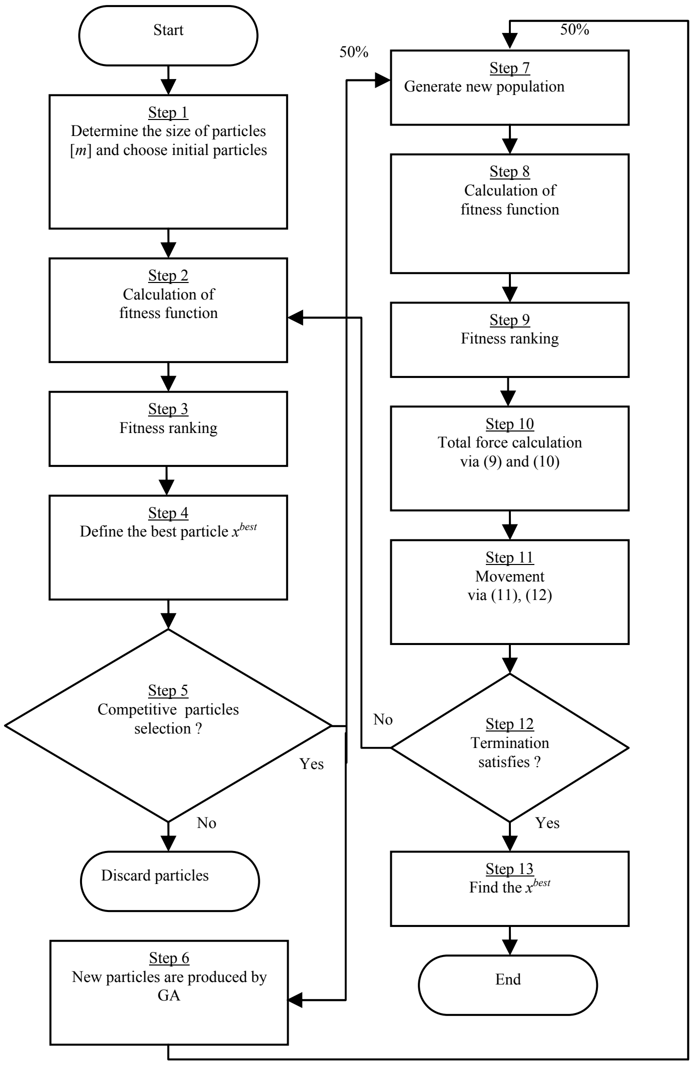
Start
Step 1
Determine the size of particles [m] and choose initial particles
Step 2
Calculation of fitness function
Step 3
Fitness ranking
Step 4
Define the best particle x^{best}
Step 5
Competitive particles selection ?
Yes
No
Discard particles
Step 6
New particles are produced by GA
50%
50%
Step 7
Generate new population
Step 8
Calculation of fitness function
Step 9
Fitness ranking
Step 10
Total force calculation via (9) and (10)
Step 11
Movement via (11), (12)
Step 12
Termination satisfies ?
No
Yes
Step 13
Find the x^{best}
End

Figure 3: Descriptions of learning algorithm IEMGA.

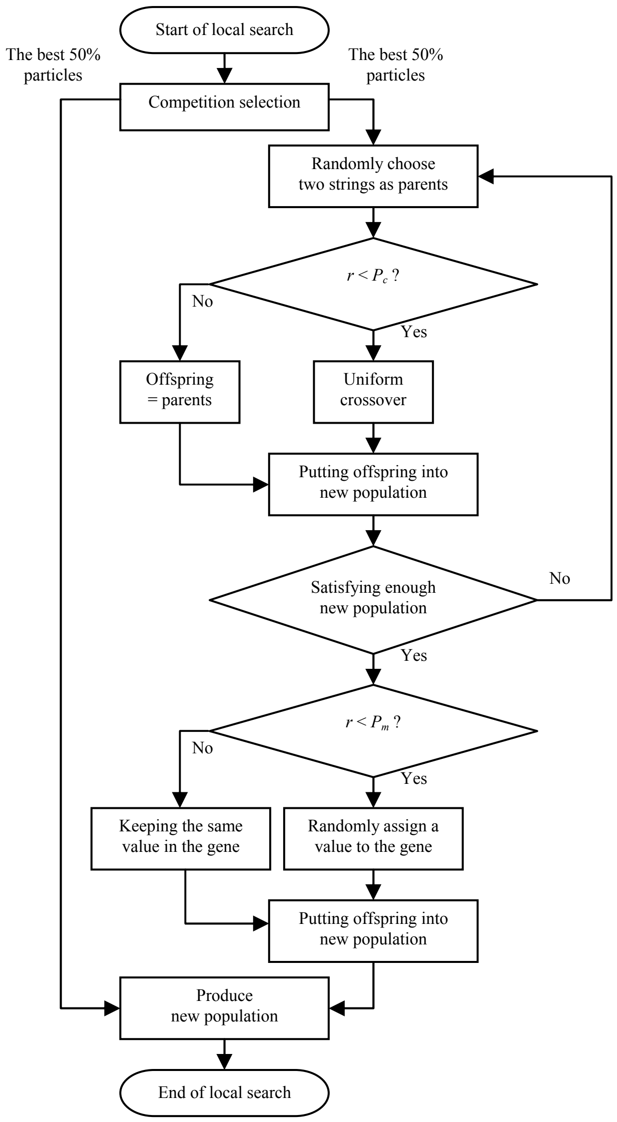
Start of local search
The best 50% particles
The best 50% particles
Competition selection
Randomly choose two strings as parents
$r < P_c$?
No
Yes
Offspring = parents
Uniform crossover
Putting offspring into new population
Satisfying enough new population
No
Yes
$r < P_m$?
No
Yes
Keeping the same value in the gene
Randomly assign a value to the gene
Putting offspring into new population
Produce new population
End of local search

Figure 4: Flow description for IEMGA local search.

A.1.2 Crossover

Based on the pre-assigned crossover rate, a certain amount of pairs of chromosomes are randomly chosen to execute the crossover operation. During this operation, one chromosome in a pair will exchange parts of its sub-sequence with the other. Generally, the common used methods are [7-8, 29, 35]:

(1) One-Point Crossover
(2) Two-Point Crossover
(3) Uniform Crossover.

In this study, we select the uniform crossover method

A.1.3 Mutation

The use of higher mutation rate P_m may be somewhat helpful in introducing gene structures that have not been searched again [7-8, 14, 30]. Randomly produce a floating-point number r between 0 and 1 for each chromosome, if $r < P_m$, the chromosome must be mutated.

Using above modifications, simple comparison results of EM and IEMGA is shown in Table 1. Obviously, the proposed IEMGA algorithm obtains a smaller computation time than the traditional EM. IEMGA only uses 2.14 seconds (about 5.2% of EM algorithm) in a generation calculation. Besides, the competition selection improve the computation efficiency of EM algorithm.

Table 1: Computation time comparison of EM and IEMGA algorithms.

Algorithm	Total time (sec)	Local search time (sec)	Percentage of local search
EM	41.07	39.36	95.84 %
IEMGA	2.14	1.65	46.45 %

IV. SIMULATION RESULTS

In this section, an example for nonlinear system control using RFNN is presented to show the performance of IEMGA. Consider the tracking control of two-input-two-output nonlinear system [14, 36]

$$y_{p1}(k+1)=0.5\cdot\left[\frac{y_{p1}(k)}{1+y_{p2}^2(k)}+u_1(k-1)\right] \tag{13}$$

$$y_{p2}(k+1)=0.5\cdot\left[\frac{y_{p1}(k)\cdot y_{p2}(k)}{1+y_{p2}^2(k)}+u_2(k-1)\right] \tag{14}$$

Note that system state $\mathbf{y_p}$=[y_{p1} y_{p2}] and tracking trajectory vector is $\mathbf{y_r}$=[y_{r1} y_{r2}], where $y_{r1}(k)=\sin(k\pi/45)$ and $y_{r2}(k)=\cos(k\pi/45)$. The inputs of RFNN controller are $\mathbf{y_p}$ and $\mathbf{y_r}$ and the outputs are u_1 and u_2. The system control scheme is shown in Fig. 9. Thus, the corresponding RMSE function of tracking error is defined

RMSE:

$$\left[\frac{1}{N}\sum_{k=1}^{N}(y_{r1}(k+1)-y_{p1}(k+1))^2+(y_{r2}(k+1)-y_{p2}(k+1))^2\right]^{\frac{1}{2}} \tag{15}$$

where N is the number of time step. The fitness function is defined as 1/(1+RMSE).

To show the effectiveness and efficiency of IEMGA, EM, and GA are used to have comparison results.

For IEMGA, EM, and GA algorithm, the following parameters are chosen

- Total generation number (or epoch): 20
- Population size (or particles): 30
- Crossover rate of GA: 0.8
- Mutation rate: 0.03

The RFNN's initial parameters m,σ,θ,ω are chosen randomly between [-1, 1] and the network structure is

- Network structure: 4-20-10-2
- Parameter number of RFNN: 70
- Rule number of RFNN: 5

Simulation results are shown in Figs. 5-6 and Table 2. Figure 5 shows the system trajectories after 20 training cycles (solid line: desired trajectory; dashed line: system actual output). Comparison results of RMSE between IEMGA and other algorithms are shown in Fig. 6 (dashed line: GA, dash-dotted line: EM, and solid line: IEMGA). Obviously, the IEMGA algorithm has better performance in RMSE than the EM algorithm and GA algorithm. The computation time of EM and IEMGA are 614.182 seconds and 25.138, respectively. IEMGA enhances the performance of EM in computation effort. Table 2 shows the comparison results of RMSE and computation time. From Table 2, IEMGA algorithm has best approximation result (Mean RMSE: 0.5814). We see that the best, worst, and mean RMSEs of IEMGA are smaller than those of EM and GA. In addition, the corresponding computation time comparison using MATLAB is also shown in Table 2. Obviously, we can conclude that the IEMGA does reduce the computation complexity of multiple particles optimization.

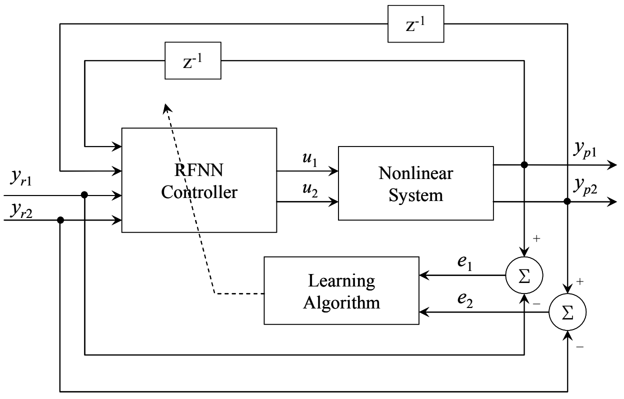

Figure 5: Dynamic system control configuration with RFNN controller.

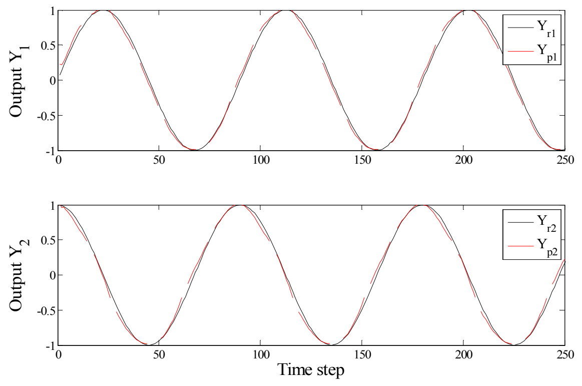

Figure 5: System trajectories after 20 training cycles of Example 1: (solid line: desired trajectory; dashed line: system actual output).

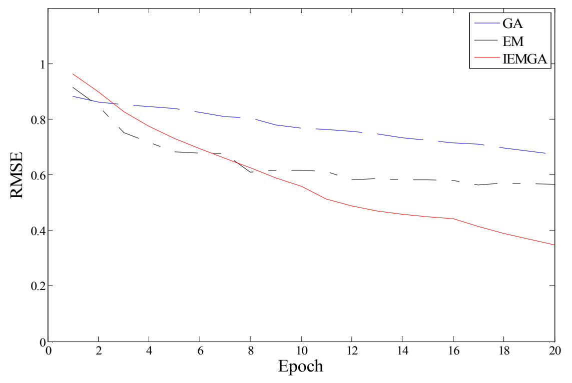

Figure 6: Comparison results of tracking error RMSE for Example 1: (dashed line: GA, dash-dotted line: EM, and solid line: IEMGA).

Table 2: Comparison results of Example 1: RMSE and computation time.

Algorithm	Best RMSE	Worst RMSE	Mean RMSE
GA	0.67274	0.92351	0.7851
EM	0.56687	0.75151	0.6379
IEMGA	0.34856	0.68149	0.5814

V. Conclusion

In this paper, we propose novel hybrid learning algorithm IEMGA for training the RFNN system. For gradient information free system, IEMGA is proposed to treat the optimization problem. IEMGA consists of EM and GA to reduce the computation complexity of EM. The major modification from EM algorithm is the randomly neighborhood local search is replaced by GA. In addition the competitive selection concept is adopted for reducing the computation. The advantages of GA are multiple searches, global optimization. IEMGA is used to develop the update laws of RFNN for nonlinear system control. Illustration example of nonlinear system control is proposed to show that IEMGA have the ability of gradient information free and it is capable of decreasing the computational complexity.

References

[1] S. I. Birbil and S. C. Fang, “An Electromagnetism-like Mechanism for Global Optimization,” *Journal of Global Optimization*, Vol. 25, No.3, pp. 263-282, 2003.

[2] S. I. Birbil, S. C. Fang, and R. L. Sheu, “On the Convergence of A Population-based Global Optimization Algorithm,” *Journal of Global Optimization*, Vol. 30, No.2, pp. 301-318, 2004.

[3] S. I. Birbil and O. Feyzioglu, “A Global Optimization Method for Solving Fuzzy Relation Equations,” *Lecture Notes in Artificial Intelligence*, Vol. 2715, pp. 718-724, 2003.

[4] P. C. Chang, S. H. Chen, and C. Y. Fan, “A Hybrid Electromagnetism-like Algorithm in Single Machine Scheduling Problems with Learning Effect,” *Appear in Expert Systems with Appl.*, Vol. 39, No.3, 2008.

[5] Y. C. Chen and C. C. Teng, “A Model Reference Control Structure Using a Fuzzy Neural Network,” *Fuzzy Sets and Systems*, Vol. 73, No.3, pp. 291-312, 1995.

[6] M. Clerc and J. Kenney, “The Particle Swarm-explosion, Stability, and Convergence in A Multidimensional Complex Space,” *IEEE Trans. on Evolutionary Computation*, Vol. 6, No.1, pp. 58-73, 2002.

[7] W. A. Farag, V. H. Quintana, and L. T. Germano, “A Genetic-based Neuro-fuzzy Approach for Modeling and Control of Dynamical Systems,” *IEEE Trans. on Neural Networks*, Vol. 9, No. 5, pp. 756-767, 1998.

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

[9] V. Gorrini and H. Bersini, “Recurrent Fuzzy Systems,” *In Proc. IEEE Int. Conf. Fuzzy Systems*, Vol. 1, pp. 193-198, June, 1994.

[10] V. G. Gudise and G. K. Venayagamoorthy, “Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithm for Neural Networks,” *IEEE Swarm Intelligence Symp.*, pp. 110-117, April, 2003.

[11] G. Hong and M. Z. Yuan “Immune Algorithm,” *Proc. of the 4th World Congress on Intelligent Control and Automation*, Vol. 3, pp. 1784-1788, June, 2002.

[12] S. Horikawa, T. Furuhashi, and Y. Uchikawa, “On Fuzzy Modeling Using Fuzzy Neural Networks with the Back-propagation Algorithm,” *IEEE Trans. on Neural Networks*, Vol. 3, No. 5, pp. 801-806, 1992.

[13] C. F. Juang, “A TSK-type Recurrent Fuzzy Network for Dynamic Systems Processing by Neural Network and Genetic Algorithms,” *IEEE Trans. on Fuzzy Systems*, Vol. 10, No. 2, pp. 155-170, 2002.

[14] C. F. Juang, “A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design,” *IEEE Trans. on Systems, Man, Cybernetics- Part: B*, Vol. 34, No.2, pp. 997-1006, 2004.

[15] D. H. Kim, “Parameter Tuning of Fuzzy Neural Networks by Immune Algorithm,” *IEEE Int. Conf. on Fuzzy Systems*, Vol. 1, pp. 408-413, May, 2002.

[16] C. H. Lee, “Stabilization of Nonlinear Nonminimum Phase Systems: An Adaptive Parallel Approach Using Recurrent Fuzzy Neural Network,” *IEEE Trans. on Systems, Man, Cybernetics- Part: B*, Vol. 34, No. 2, pp. 1075-1088, 2004.

[17] C. H. Lee and F. K. Chang, “Recurrent Fuzzy Neural Controller Design for Nonlinear Systems Using Electromagnetism-like Algorithm,” *Far East Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 1, No. 1, pp. 5-22, 2008.

[18] C. H. Lee, F. K. Chang, and C. W. Chen, “A Modified Electromagnetism-like Algorithm for Training Neural Fuzzy Systems in Control Applications,” *CACS Int. Automatic Control Conf.*, Taichung, Taiwan, Nov., 2007.

[19] C. H. Lee and M. H. Chiu, “Adaptive Nonlinear Control Using TSK-type Recurrent Fuzzy Neural Network System,” *Lecture Notes in Computer Science*, Vol. 4491, pp. 38-44, 2007.

[20] C. H. Lee and C. C. Teng, “Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks,” *IEEE Trans. on Fuzzy Systems*, Vol. 8, No. 4, pp. 349-366, 2000.

[21] C. J. Lin and Y. C. Hsu, “Reinforcement Hybrid Evolutionary Learning for Recurrent Wavelet-based Neuro-fuzzy Systems,” *IEEE Trans. on Fuzzy Systems*, Vol. 15, No. 4, pp. 729-745, 2007.

[22] C. J. Lin, C. Y. Lee, and C. C. Chin, “Dynamic Recurrent Wavelet Network Controllers for Nonlinear System Control,” *Journal of The Chinese Institute of Engineers*, Vol. 29, No. 4, pp. 747-751, 2006.

[23] C. J. Lin and Y. J. Xu, “A Novel Evolution Learning for Recurrent Wavelet-based Neuro-fuzzy Networks,” *Soft Computing Journal*, Vol. 10, No. 3, pp. 193-205, 2006.

[24] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems*, Prentice Hall: Englewood Cliff, 1996.

[25] C. T. Lin and C. S. G. Lee, “Neural-network-based Fuzzy Logic Control and Decision System,” *IEEE Trans. on Computers*, Vol. 40, No. 12, pp. 1320-1336, 1991.

[26] F. J. Lin, R. J. Wai, and C. C. Lee, “Fuzzy Neural Network Position Controller for Ultrasonic Motor Drive Using Push-pull DC-DC Converter,” *IEE Proc.- Control Theory and Appl.*, Vol. 146, No. 1, pp. 99-107, 1999.

[27] P. A. Mastorocostas and J. B. Theocharis, “A Recurrent Fuzzy-neural Model for Dynamic System Identification,” *IEEE Trans. on Systems, Man, Cybernetics- Part: B*, Vol. 32, No. 2, pp. 176-190, 2002.

[28] J. B. Pomet, “Explicit Design of Time-varying Stabilizing Control Laws for A Class of Controllable Systems Without Drift,” *Systems and Control Letters*, Vol. 18, No. 2, pp. 147-158, 1992.

[29] M. N. H. Siddique and M. O. Tokhi, “Training Neural Networks: Backpropagation vs. Genetic Algorithms,” *Proc. of Int. J. Conf. on Neural Networks*, Vol. 4, pp. 2673-2678, 2001.

[30] M. Srinivas and L. M. Patnaik, “Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms,” *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 24, No. 4, pp. 656-667, 1994.

[31] C. S. Tsou and C. H. Kao, “Multi-objective Inventory Control Using Electromagnetism-like Meta-heuristic,” *Int. Journal of Production Research*, Vol. 1, pp. 1-16, 2007.

[32] D. F. Wang, P. Han, N. Liu, Z. Dong, and S. M. Jiao, “Modeling the Circulating Fluidized Bed Boiler Using RBF-NN Based on Immune Genetic Algorithm,” *Proc. of the First Int. Conf. on Machine Learning and Cybernetics*, Vol. 4, pp. 2121-2125, Nov., 2002.

[33] J. S. Wang and Y. P. Chen, “A Fully Automated Recurrent Neural Network for Unknown Dynamic System Identification and Control,” *IEEE Trans. on Circuits and Systems*-I, Vol. 56, No. 6, pp. 1363-1372, 2006.

[34] P. Wu, W. H. Yang, and N. C. Wei, “An Electromagnetism Algorithm of Neural Network Analysis-An Application to Textile Retail Operation,” *Journal of the Chinese Institute of Industrial Engineers*, Vol. 21, No. 1, pp. 59-67, 2004.

[35] H. G. Xu, X. Y. Wei, and M. S. Xu, “Schema Analysis of Multi-points Crossover Genetic Algorithm,” *Proc. 3rd World Congress on Intelligent Control and Automation*, Vol. 1, pp. 521-524, 2000.

[36] X. Yao, *Evolutionary Computation: Theory and Applications*, World Scientific, 1999.

[37] J. Zhang, and A. J. Morris, “Recurrent Neuro-fuzzy Networks for Nonlinear Process Modeling,” *IEEE Trans. on Neural Networks*, Vol. 10, pp. 313-326, 1999.