Efficient Cooperative Particle Swarm Optimization for TSK-Type Neural Fuzzy Systems and Its Classification Applications

Cheng-Hung Chen*

Department of Electrical Engineering, National Formosa University, No. 64, Wunhua Rd., Huwei Township, Yunlin County 632, Taiwan

Abstract: This study proposes a cooperative particle swarm optimization (CPSO) to optimize the parameters of the TSK-type neural fuzzy system (TNFS) for classification applications. The proposed CPSO uses cooperative behavior among multiple subswarms to decompose the neural fuzzy systems into rule-based subswarms, and each particle within each subswarm evolves by a specific particle swarm optimization (PSO) separately. Therefore, the CPSO can accelerate the search and increase global search capacity. Finally, the TNFS with CPSO (TNFS-CPSO) is adopted in several classification applications. Experimental results demonstrate that the proposed TNFS-CPSO method has a higher accuracy rate and a faster convergence rate than the other methods.

Keywords: Particle swarm optimization, cooperative evolution, TSK-type neural fuzzy systems, classification.

1. INTRODUCTION

In recent years, neural fuzzy systems for classification applications have become a popular research topic [1-8]. The key advantage of the neural fuzzy systems lies in the fact that it does not require a mathematical description of a system when the system is modeled. Two typical types of neural fuzzy systems are the Mamdani-type and TSK-type neural fuzzy systems. Many researchers have shown that using the TSK-type neural fuzzy systems (TNFS) superior performance in system size and learning accuracy than using the Mamdani-type neural fuzzy systems [9-10]. Furthermore, backpropagation (BP) algorithm is usually adopted to train the parameters of neural fuzzy systems to make neural fuzzy systems more adaptive and effective. BP is a powerful training technique that can be applied to neural fuzzy systems with a forward structure. Unfortunately, BP may reach the local minima very quickly and never find the global solution, because the steepest descent technique is used in BP training to minimize the error function.

The aforementioned disadvantages lead to suboptimal performance, even for a favorable neural fuzzy system. Therefore, technologies that can be used to train the system parameters and find the global solution while optimizing the overall structure are required. Accordingly, a new optimization algorithm, called particle swarm optimization (PSO), appears to be better than BP. PSO is a stochastic optimization technique developed by Kennedy and Eberhart in 1995 [11-12], inspired by the social behavior of bird flocking or fish schooling. Bird flocking has some underlying rules that enables large numbers of birds to flock synchronously, often changing direction suddenly. PSO has been successfully applied to many optimization problems [13-21].

Furthermore, an increasingly large and complex system often requires solving complex high-dimensional optimization problems. When the dimension of the problem increases, the overall performance of the system often decreases significantly. Cooperative evolution has been proven to be an effective solution to above-mentioned problems [22-24]. Cooperative evolution consists of a number of cooperative populations and processes the low-dimensional subcomponents of the original problem simultaneously. Cooperation between the subcomponents combines their strengths and builds a complete solution to the original problem.

In this study, the TNFS model with the CPSO method (TNFS-CPSO) is proposed for classification applications. The proposed CPSO learning algorithm uses cooperative evolution to decompose the neural fuzzy systems into the rule-based subswarms. Each subswarm represents a set of the single fuzzy rule, and each particle in each rule-based subswarm evolves by PSO separately. The proposed CPSO embeds cooperative evolution into PSO to accelerate the search and increase global search capacity.

This study is organized as follows. Section 2 describes the structure of the TNFS model. The proposed CPSO is presented in Section 3. In Section 4, the proposed CPSO method is evaluated, and its performances are benchmarked against other

*Address correspondence to this author at the Department of Electrical Engineering, National Formosa University, No.64, Wunhua Rd., Huwei Township, Yunlin County 632, Taiwan; Tel: +886-5-6315612;
Fax: +886-5-6315609: E-mail: chchen.ee@nfu.edu.tw

methods. Finally, conclusions on the proposed method are given in the last section.

2. STRUCTURE OF THE TSK-TYPE NEURAL FUZZY SYSTEM

This section describes the TNFS model [25]. A fuzzy logic is a knowledge-based system characterized by a set of rules that determine the relationship between the input and the output. The reasoning process is defined by means of the inference method, aggregation operators, and fuzzy connectives. The fuzzy knowledge base contains the definition of fuzzy sets, which is stored in a fuzzy database, and a collection of fuzzy rules. The definition of fuzzy sets and the collection of fuzzy rules constitute the fuzzy rule base.

Fuzzy rules are defined by their antecedents and consequents, which relate an observed input state to a desired output. Most neural fuzzy systems employ the inference method proposed by Mamdani in which the consequent parts are defined by fuzzy sets [26]. A Mamdani-type fuzzy rule has the form:

IF x_1 is A_{1j} $(m_{1j}, \sigma_{1j})$and x_2 is $A_{2j}(m_{2j}, \sigma_{2j})$…and x_n is A_{nj} (m_{nj}, σ_{nj})

THEN y' is B_j (m_j, σ_j) (1)

where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation, respectively, of the *i*th dimension and the *j*th rule node. The consequent B_j of the *j*th rule is aggregated into one fuzzy set for the output variable y'. The crisp output is obtained through defuzzification, which calculates the centroid of the output fuzzy set. In addition, the more common fuzzy inference method proposed by Mamdani and Takagi-Sugeno-Kang introduced a modified inference scheme [25]. The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator, are exactly the same. A TNFS model employs different implication and aggregation methods than the standard Mamdani model. Instead of fuzzy sets being used, the conclusion part of a rule is a linear combination of the crisp inputs, as follows:

IF x_1 is A_{1j} $(m_{1j}, \sigma_{1j})$and x_2 is $A_{2j}(m_{2j}, \sigma_{2j})$…and x_n is A_{nj} (m_{nj}, σ_{nj})

THEN $y'=w_{0j}+w_{1j}x_1+\ldots+w_{nj}x_n$ (2)

where m_{ij} and σ_{ij} represent a Gaussian membership function with mean and deviation, respectively, of the *i*th dimension and the *j*th rule node. Since the consequent of a rule is crisp, the defuzzification step becomes obsolete in the TSK inference scheme. Instead, the model output is computed as the weighted average of the crisp rule outputs. This computation is less expensive than calculating the center of gravity.

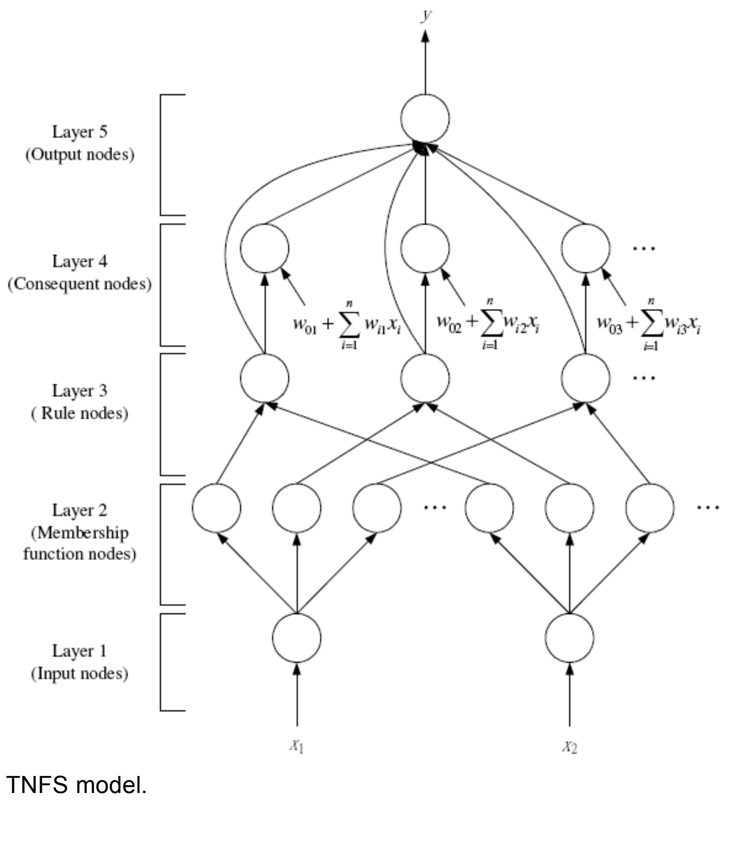
y
Layer 5 (Output nodes)
Layer 4 (Consequent nodes)
Layer 3 (Rule nodes)
Layer 2 (Membership function nodes)
Layer 1 (Input nodes)
$w_{01}+\sum_{i=1}^{n} w_{i1}x_i$
$w_{02}+\sum_{i=1}^{n} w_{i2}x_i$
$w_{03}+\sum_{i=1}^{n} w_{i3}x_i$
x_1
x_2

Figure 1: The structure of the TNFS model.

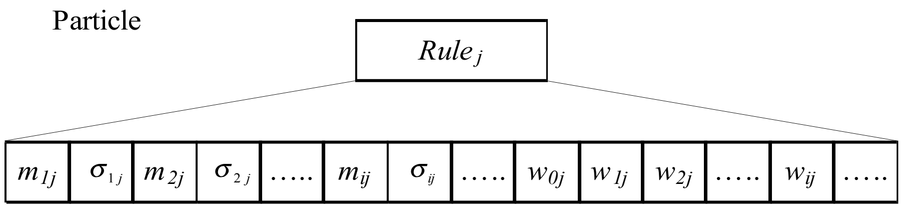

Figure 2: Coding a fuzzy rule into a particle in the proposed CPSO method.

3. COOPERATIVE PARTICLE SWARM OPTIMIZATION FOR THE TSK-TYPE NEURAL FUZZY SYSTEM

This section describes the proposed CPSO to optimize the parameters of the TNFS model. In the proposed method, cooperative evolution is used to effectively decompose the neural fuzzy systems into rule-based subswarms and evolve these subswarms cooperatively. Cooperation evolution among the subswarms is responsible for combining their information and building the complete neural fuzzy system. Furthermore, each particle in each subswarm evolves separately using PSO. The foremost step in CPSO is the coding of a fuzzy rule into a particle. The coding of the parameters of a fuzzy rule into a particle is shown in Figure **2**, where m_{ij} and σ_{ij} are the mean and standard deviation of the *i*th input variable and the *j*th rule of the Gaussian membership function, respectively, and w_j represents the corresponding link weight of the consequent part that is connected to the *j*th rule node. In this study, a real number represents the position of a particle. A flowchart describing this process is presented in Figure **3**. The proposed CPSO process is described step-by-step below.

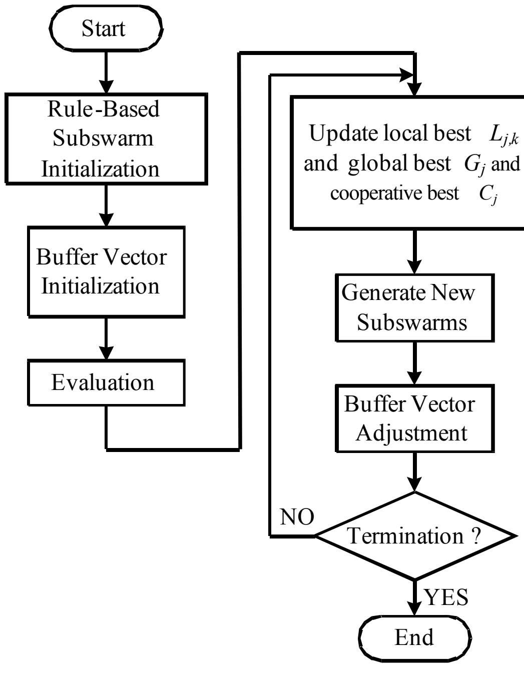
Start
Rule-Based Subswarm Initialization
Buffer Vector Initialization
Evaluation
Update local best $L_{j,k}$ and global best G_j and cooperative best C_j
Generate New Subswarms
Buffer Vector Adjustment
Termination ?
NO
YES
End

Figure 3: Flowchart of the proposed CPSO method.

Step 1: Subswarm Initialization

Before the CPSO method is applied, every position $x_{j,k}$ must be created randomly in the range [0, 1] in each subswarm, where *j*=1, 2,…, *R* represents the *j*th subswarm and *k*=1, 2, …, *ps* represents the *k*th particle.

Step 2: Buffer Vector Initialization

In the rule-based subswarm, each particle represents a fuzzy rule. To evaluate each particle, the particle is added to the buffer vector (i.e., best neural fuzzy system) with the other particles of the subswarms. The initial buffer vector is composed of randomly selected particles from each subswarm.

Step 3: Evaluation

There is a conspicuous problem in evaluating the particles using the fitness function because they come from different subswarms. This problem can be solved by defining a buffer vector, an information sharing mechanism in the form of a public memory area that stores particles with the best performances. The buffer vector is defined as follows:

$$buffer = [S_{1,1},...,S_{D,1},S_{1,2},...,S_{D,2}, \\ ...,S_{1,j},...,S_{D,j},...,S_{1,R},...,S_{D,R}] \quad (3)$$

The dimension of the buffer vector is $D \times R$, where *R* represents the number of fuzzy rules and $[S_{1,j},...,S_{D,j}]$ represents the contributor in the *j*th subswarm.

The *k*th particle of the *j*th subswarm is evaluated using the buffer vector to replace the corresponding contributor with the evaluated particle. Therefore, the evaluated particle is defined as follows:

$$[S_{1,1},..., S_{D,1}, S_{1,2},..., S_{D,2},..., \underbrace{x_{j,1,k},..., x_{j,D,k}}_{\text{particle } x_{j,k}},..., S_{1,R},..., S_{D,R}] \quad (4)$$

where $x_{j,k}$ is the *k*th evaluated particle of the *j*th subswarm.

In this study, a fitness function is adopted to evaluate the performance of the neural fuzzy system. The fitness function is defined as follows

$$f(x)=\frac{1}{1+\sqrt{\frac{1}{N_t}\sum_{s=1}^{N_t}(y_s-\overline{y}_s)^2}} \tag{5}$$

where y_s is the model output of the *s*th data, $\overline{y}_s$ is the desired output of the *s*th data, and N_t is the number of the training data.

Step 4: Update local best $L_{j,k}$, global best G_j, and cooperative best C_j

The local best position $L_{j,k}$ is the best previous position that yielded the best fitness value of the *j*th subswarm of the *k*th particle and the global best position G_j is generated by the whole local best position. In this step, the first step updates the local best position. Compare the fitness value of each current particle with that of its local best position. If the fitness value of the current particle exceeds those of its local best position, then the local best position is replaced with the position of the current particle. The second step updates the global best position. Compare the fitness value of all particles in their local best positions with that of the particle in the global best position. If fitness value of the particle in the local best position is better than those of the particles in the global best position, then the global best position is replaced with the current local best position.

$$\begin{aligned} L_{j,k} &= \begin{cases} x_{j,k}, & \text{if } F\ (x_{j,k})\ < F\ (L_{j,k}) \\ L_{j,k}, & \text{if } F\ (x_{j,k})\ \geq F\ (L_{j,k}) \end{cases} \\ G_j &= \arg\max_{L_{j,k}} F(L_{j,k}), \qquad 1 \leq k \leq ps \end{aligned} \tag{6}$$

The third step updates the cooperative best position. Compare the fitness values of all composed neural fuzzy systems and the best neural fuzzy system. If the fitness value of one of all composed neural fuzzy systems exceeds those of the best neural fuzzy system, then the best neural fuzzy system is replaced with the composed neural fuzzy system in which the corresponding rule is the cooperative best (*Cbest*).

Step 5: Generate new subswarms using $L_{j,k}$, G_j and C_j

The step updates velocity and position of each particle to generate the new subswarms using Eqs. (7) and (8).

$$\begin{aligned} v_{j,k}(t+1) &= \omega\cdot v_{j,k}(t)+\phi_1\cdot Rand()\cdot(L_{j,k}-x_{j,k})+\phi_2\cdot \\ &Rand()\cdot(G_j-x_{j,k})+\phi_3\cdot Rand()\cdot(C_j-x_{j,k}) \end{aligned} \tag{7}$$

$$x_{j,k}(t+1)=x_{j,k}(t)+v_{j,k}(t+1) \tag{8}$$

where ω is the coefficient of inertia, ϕ_1 is the cognitive study, ϕ_2 is the society study, ϕ_3 is the group study, and $Rand()$ is generated from a uniform distribution in the range [0, 1].

Step 6: Buffer Vector Adjustment

The buffer vector needs to be adjusted to keep the best neural fuzzy system during the evolution process. This process is performed by comparing the fitness value of the current particle to that of the buffer vector. If the fitness value of the current particle exceeds the fitness value of the buffer vector (i.e., $f\left(x_{j,k}\right) > f\left(buffer\right)$), then the corresponding contributor of the buffer vector is replaced by the current particle.

5. ILLUSTRATIVE EXAMPLES

In this section, the performance of the TNFS model with the proposed CPSO method is evaluated using two better-known benchmark data sets for classification problems. The first example uses the Iris data and the second example uses the Wisconsin breast cancer data. The two benchmark data sets are available from the University of California, Irvine, *via* an anonymous ftp address ftp://ftp.ics.uci.edu/pub/machine-learning-databases. In the following simulations, the parameters and number of training epochs were based on the desired accuracy. In short, the trained TNFS with CPSO was stopped once its high learning efficiency was demonstrated. Table **1** presents the initial parameters of the proposed CPSO in the three classification problems.

Table 1: Initial Parameters before Learning

Parameter	Value
Subswarm Size	100
Maximum Number of Generation	2000
ϕ_1 , ϕ_2	1
ϕ_3	2
ω	0.4
Coding Type	Real Number

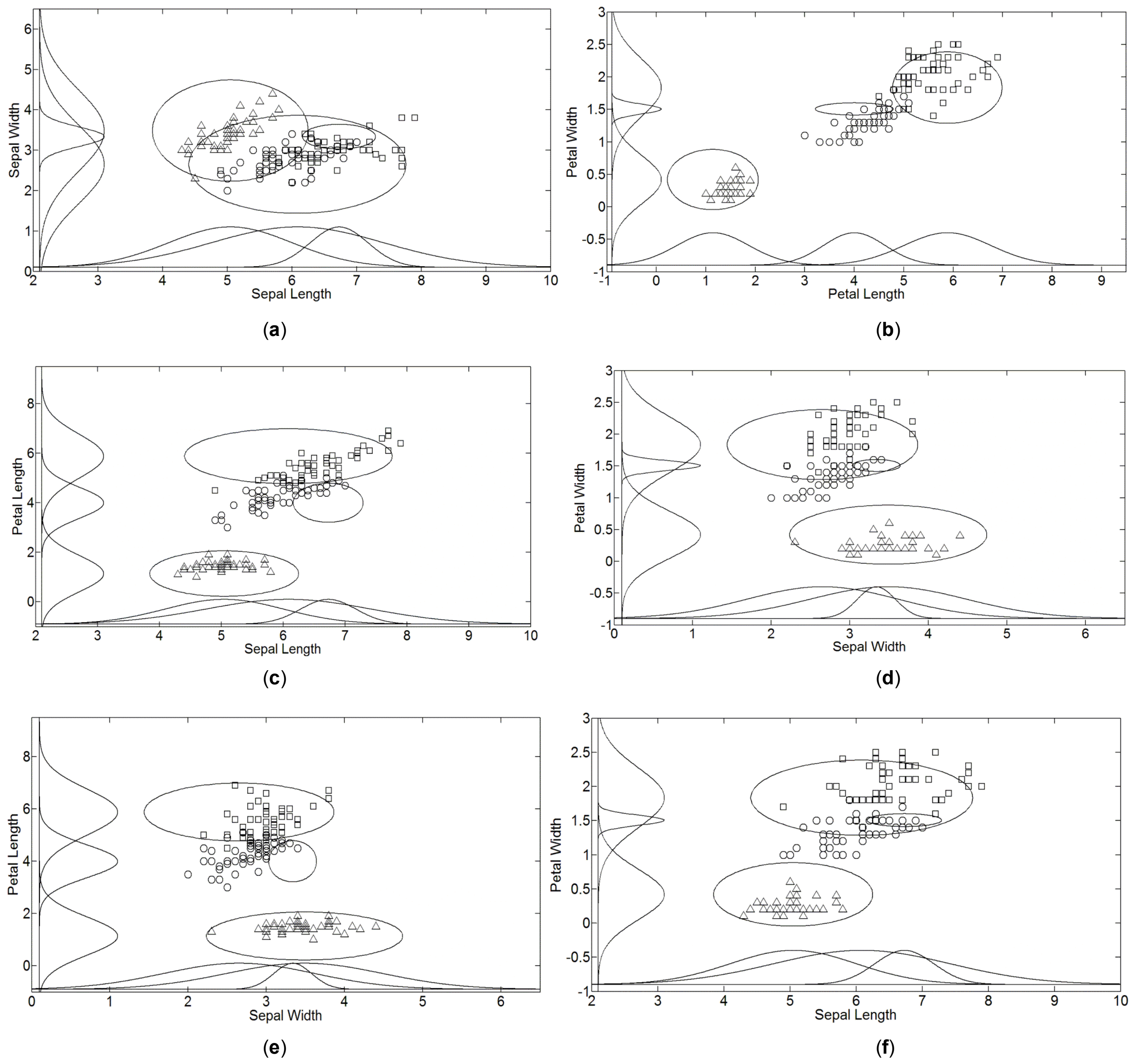

Figure 4: The distribution of input training patterns and final assignment of three rules. (**a**) For the *Sepal Length* and *Sepal Width* dimensions. (**b**) For the *Petal Length* and *Petal Width* dimensions. (**c**) For the *Sepal Length* and *Petal Length* dimensions. (**d**) For the *Sepal Width* and *Petal Width* dimensions. (**e**) For the *Sepal Width* and *Petal Length* dimensions. (**f**) For the *Sepal Length* and *Petal Width* dimensions.

5.1. Iris Data Classification

The Fisher-Anderson iris data consists of four input measurements, sepal length (*sl*), sepal width (*sw*), petal length (*pl*), and petal width (*pw*), on 150 specimens of the iris plant. Three species of iris were involved, *Iris Sestosa*, *Iris Versiolor* and *Iris Virginica*, and each species contains 50 instances.

In the Iris data experiment, 25 instances with four features from each species were randomly selected as the training set (i.e., a total of 75 training patterns were used as the training data set) and the remaining instances were used as the testing set. Once the TNFS model was trained, all 150 test patterns of the Iris data were presented to the trained TNFS model, and the re-substitution error was computed. In this example, three fuzzy rules are adopted. The learning proceeded for 2000 generations, and was repeated thirty runs. After 2000 generations, the average fitness value was 0.9424. Figure **4** (**a-f**) show the distribution of the training pattern and the final assignment of the fuzzy rules (i.e., distribution of input membership functions). Since the region covered by a Gaussian membership function is unbounded, in Figure **4** (**a-f**), the boundary of each ellipse represent a rule with a firing strength of 0.5.

In this example, the PSO [12], RPSO [27], and LPSO [28] methods were applied to the same problem to show the effectiveness and efficiency of the TNFS model with the proposed CPSO learning method. In the PSO, RPSO, and LPSO, the cognitive coefficient ϕ_1 was set to 2, the society coefficient ϕ_2 was set to 2, and the population size was set to 200. The coefficient ω of PSO was set to 0.4, the maximal and minimal weights of LPSO are set to 0.9 and 0.4, respectively. We compared the testing accuracy of our proposed method with that of other methods – TNFS-PSO, TNFS-RPSO, and TNFS-LPSO. Thirty experiments were used. These experiments calculated the classification accuracy and the values of the average produced on the testing set using the TNFS-PSO method, the TNFS-RPSO method, the TNFS-LPSO method, and the proposed TNFS-CPSO method.

During the learning phase, the learning curves from the proposed TNFS-CPSO method, the TNFS-LPSO method, the TNFS-RPSO method, and the TNFS-PSO method are shown in Figure **5**. Table **2** shows that the average classification accuracy of the TNFS-CPSO method in high accuracy was better than that of other methods.

5.2. Wisconsin Breast Cancer Diagnostic Data

The Wisconsin breast cancer diagnostic data set contains 699 patterns distributed into two output classes, “benign” and “malignant.” Each pattern consists of nine input features: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. 458 patterns are in the benign class and the other 241 patterns are in the malignant class. Since there were 16 patterns containing missing values, we used 683 patterns to evaluate the performance of the proposed TNFS-CPSO method. To compare the performance with other methods, we used half of the 683 patterns as the training set and the remaining patterns as the testing set.

Experimental conditions were the same as the previous experiment. We also used half of the original data patterns as the training data (randomly selected) and the remaining patterns as the testing data. For training, the training patterns were randomly chosen, and the remaining patterns were used for testing.

This example, as subsection 5.1, compares the performance of the TNFS-CPSO with that of other methods. In the PSO, RPSO and LPSO methods, the parameters are the same as in subsection 5.1. Thirty experiments also were used. These experiments calculated the classification accuracy and the values of

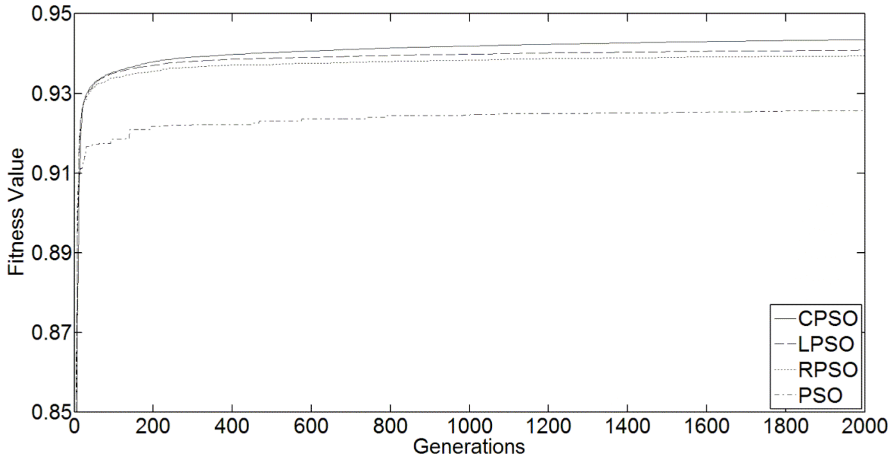

Figure 5: Learning curves of the proposed CPSO method, the LPSO method, the RPSO method, and the PSO method.

Table 2: Classification Accuracy Using Various Methods for the Iris Data

Method	TNFS-PSO	TNFS-RPSO	TNFS-LPSO	TNFS-CPSO
Mean (%)	91.33	94.76	95.82	97.33
Std Dev	3.3356	2.0085	1.9102	1.1087

the average produced on the testing set by the TNFS-PSO method, the TNFS-RPSO method, the TNFS-LPSO method, and the proposed TNFS-CPSO method. During the supervised learning phase, 2000 epochs of training were performed. Figure **6** shows the membership functions for each input feature. The learning curves from the proposed TNFS-CPSO method, the TNFS-LPSO method, the TNFS-RPSO method, and the TNFS-PSO method are shown in Figure **7**. The performance of the TNFS-CPSO method is better than the performance of all other methods. Table **3** shows that the average classification accuracy of the TNFS-CPSO method was better than that of other methods.

6. CONCLUSION

This study proposes a CPSO method for a TNFS model in classification problems. The major novelty of the proposed CPSO learning algorithm uses the rule-based subswarms to allow that each particle in each subswarm evolves separately using a specific PSO for constructing the TNFS-CPSO method. Furthermore, the proposed CPSO embeds cooperative evolution into PSO to accelerate the search and increase global search capacity. Three examples showed that the proposed TNFS-CPSO method improves the system performance in terms of a fast learning convergence, and a high correct classification rate.

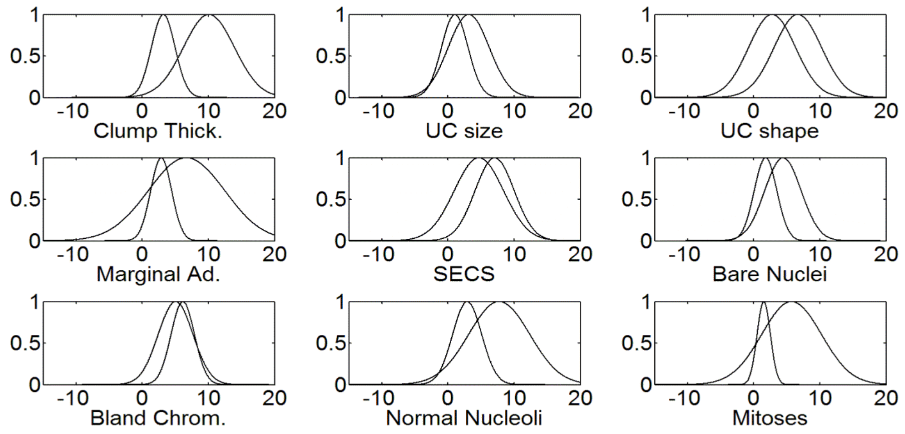

Figure 6: Input membership functions for breast cancer classification.

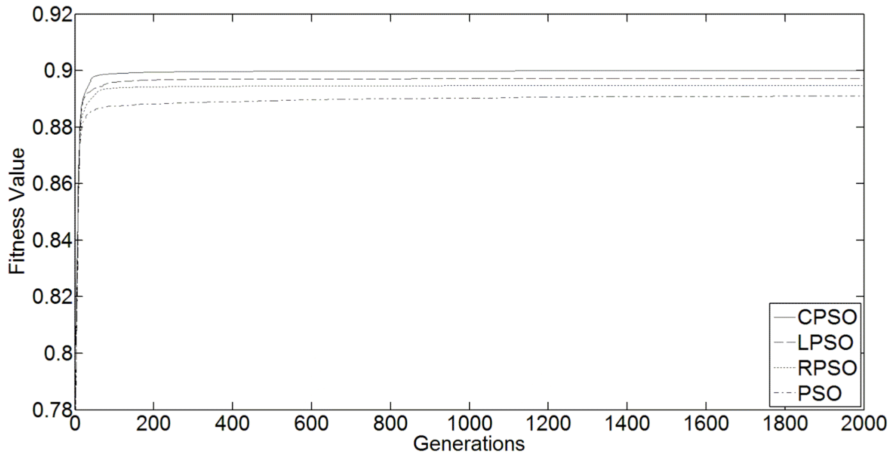

Figure 7: Learning curves from the proposed CPSO method, the LPSO method, the RPSO method, and the PSO method.

Table 3: Classification Accuracy for the Wisconsin Breast Cancer Diagnostic Data

Method	TNFS-PSO	TNFS-RPSO	TNFS-LPSO	TNFS-CPSO
Mean (%)	91.8	94.51	95.61	97.39
Std Dev	0.0154	0.0127	0.0105	0.0087

REFERENCES

[1] Duda PO, Hart PE. Pattern Classification and Scene Analysis. New York: Wiley 1973.

[2] Nauck D, Kruse R. A neuro-fuzzy method to learn fuzzy classification rules from data. Fuzzy Set Syst 1997; 89(3): 277-288. http://dx.doi.org/10.1016/S0165-0114(97)00009-2

[3] Ishibuchi H, Nakashima T, Murata T. Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. IEEE T Syst Man Cy B 1999; 29(5): 601-618. http://dx.doi.org/10.1109/3477.790443

[4] Kasabov N. Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. IEEE T Syst Man Cy B 2001; 31(6): 902-918. http://dx.doi.org/10.1109/3477.969494

[5] Ajiboye AB, Weir RF. A heuristic fuzzy logic approach to EMG pattern recognition for multifunctional prosthesis control. IEEE T Neur Sys Reh 2005; 13(3): 280-291. http://dx.doi.org/10.1109/TNSRE.2005.847357

[6] Lin CJ, Xu YJ. Efficient reinforcement learning through dynamic symbiotic evolution for TSK-type fuzzy controller design. Int J Gen Syst 2005; 34(5): 559-578. http://dx.doi.org/10.1080/03081070500132377

[7] Ducange P, Lazzerini B, Marcelloni F. Multi-objective genetic fuzzy classifiers for imbalanced and cost-sensitive datasets. Soft Comput 2010; 14(7): 713-728. http://dx.doi.org/10.1007/s00500-009-0460-y

[8] He C, Ye Y. Evolution computation based learning algorithms of polygonal fuzzy neural networks. Int J Intell Syst 2011; 26(4): 340-352. http://dx.doi.org/10.1002/int.20469

[9] Jang JSR. ANFIS: Adaptive-network-based fuzzy inference system. IEEE T Syst Man Cy 1993; 23(3): 665-685. http://dx.doi.org/10.1109/21.256541

[10] Juang CF, Lin CT. An online self-constructing neural fuzzy inference network and its applications. IEEE T Fuzzy Syst 1998; 6(1): 12-31. http://dx.doi.org/10.1109/91.660805

[11] Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: Proc. 6th Int Symp MicroMach Hum Sci 1995; pp 39-43.

[12] Kennedy J, Eberhart R. Particle swarm optimization. In: Proc IEEE Int Conf Neural Netw 1995; pp 1942-1948.

[13] Yoshida H, Kawata K, Fukuyama Y, Takayama S, Nakanishi Y. A particle swarm optimization for reactive power and voltage control considering voltage security assessmen. IEEE T Power Syst 2000; 15(4): 1232-1239. http://dx.doi.org/10.1109/59.898095

[14] Gaing ZL. A particle swarm optimization approach for optimum design of PID controller in AVR system. IEEE T Energy Conver 2004; 19(2): 384-391. http://dx.doi.org/10.1109/TEC.2003.821821

[15] Feng HM, Chen CY, Ye F. Adaptive hyper-fuzzy partition particle swarm optimization clustering algorithm. Cybernet Syst 2006; 37(5): 463-479. http://dx.doi.org/10.1080/01969720600683429

[16] Huang VL, Suganthan PN, Liang JJ. Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems. Int J Intell Syst 2006; 21(2): 209-226. http://dx.doi.org/10.1002/int.20128

[17] Cai X, Zhang N, Venayagamoorthy GK, Wunsch DC. Time series prediction with recurrent neural networks trained by a hybrid PSO-EA algorithm. Neurocomputing 2007; 70(13-15): 2342-2353. http://dx.doi.org/10.1016/j.neucom.2005.12.138

[18] Song Y, Chen Z, Yuan Z. New chaotic PSO-based neural network predictive control for nonlinear process. IEEE T Neur Net Lear 2007; 18(2): 595-601. http://dx.doi.org/10.1109/TNN.2006.890809

[19] Zhang JR, Zhang J, Lok TM, Lyu MR. A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training. Appl Math Comput 2007; 185(2): 1026-1037. http://dx.doi.org/10.1016/j.amc.2006.07.025

[20] del Valle Y, Venayagamoorthy GK, Mohagheghi S, Hernandez JC, Harley RG. Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE T Evol Comput 2008; 12(2): 171-195. http://dx.doi.org/10.1109/TEVC.2007.896686

[21] Lin CJ, Peng CC. Chord recognition using neural networks based on particle swarm optimization. Cybernet Syst 2011; 42(4): 264-282. http://dx.doi.org/10.1080/01969722.2011.583597

[22] Potter MA, De Jong KA. A cooperative coevolutionary approach to function optimization. In: Proc. of the Third Conference on Parallel Problem Solving from Nature 1994; pp 249-257. http://dx.doi.org/10.1007/3-540-58484-6_269

[23] Potter MA, De Jong KA. Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evol Comput 2000; 8(1): 1-29. http://dx.doi.org/10.1162/106365600568086

[24] Yang Z, Tang K, Yao X. Large scale evolutionary optimization using cooperative coevolution. Inform Sciences 2008; 178(15): 2985-2999. http://dx.doi.org/10.1016/j.ins.2008.02.017

[25] Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control. IEEE T Syst Man Cybern 1985; 1(1): 116-132. http://dx.doi.org/10.1109/TSMC.1985.6313399

[26] Lin CT, Lee CSG. Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System. NJ: Prentice-Hall 1996.

[27] Shi Y, Eberhart RC. A modified particle swarm optimizer. In: Proc. of IEEE International Conference on Evolutionary Computation, Anchorage, AK, 1998; pp 69-73.

[28] Eberhart RC, Shi Y. Tracking and optimizing dynamic systems with particle swarms. In: Proc. of IEEE Congress on Evolutionary Computation, Seoul, Korea, 2001; pp 94-97.

Received on 05-12-2014 Accepted on 18-12-2014 Published on 02-04-2015

DOI: http://dx.doi.org/10.15377/2409-5761.2014.02.01.6