Automatic Design of Hierarchical Takagi–Sugeno Type Fuzzy Systems Using Evolutionary Algorithms

Yuehui Chen, *Member, IEEE*, Bo Yang, Ajith Abraham, *Member, IEEE*, and Lizhi Peng

***Abstract*—This paper presents an automatic way of evolving hierarchical Takagi–Sugeno fuzzy systems (TS-FS). The hierarchical structure is evolved using probabilistic incremental program evolution (PIPE) with specific instructions. The fine tuning of the *IF–THEN* rule's parameters encoded in the structure is accomplished using evolutionary programming (EP). The proposed method interleaves both PIPE and EP optimizations. Starting with random structures and rules' parameters, it first tries to improve the hierarchical structure and then as soon as an improved structure is found, it further fine tunes the rules' parameters. It then goes back to improve the structure and the rules' parameters. This loop continues until a satisfactory solution (hierarchical TS-FS model) is found or a time limit is reached. The proposed hierarchical TS-FS is evaluated using some well known benchmark applications namely identification of nonlinear systems, prediction of the Mackey–Glass chaotic time-series and some classification problems. When compared to other neural networks and fuzzy systems, the developed hierarchical TS-FS exhibits competing results with high accuracy and smaller size of hierarchical architecture.**

***Index Terms*—Classification, evolutionary programming, hierarchical Takagi–Sugeno fuzzy systems (TS-FS) model, probabilistic incremental program evolution algorithm, system identification, time-series prediction.**

I. Introduction

FUZZY inference systems [1], [2], [31] have been successfully applied to a number of scientific and engineering problems during recent years. The advantage of solving complex nonlinear problems by utilizing fuzzy logic methodologies is that the experience or expert's knowledge described as the fuzzy rule base can be directly embedded into the system for dealing with the problems. Many efforts have been made to enhance systematic design of fuzzy logic systems [4]–[8], [46], [51]. Some researches focus on automatically finding the appropriate structure and parameters of fuzzy logic systems by using genetic algorithms [5], [8], [46], evolutionary programming [7], tabu search [10], and so on. There are many research works focusing on partitioning of the input space, to determine the fuzzy rules and parameters evolved in the fuzzy rules for a single fuzzy system [34], [35]. As it is well known, the curse-of-dimensionality is an unsolved problem in the fields of fuzzy and/or neurofuzzy systems [50].

Some of the problems mentioned above are partially solved by several researchers working in the hierarchical fuzzy systems domain [10]–[18], [33], [53], [54]. Torra [3] has summarized the related recent researches. As a way to overcome the curse-of-dimensionality, it was suggested by Brown *et al.* [18] to arrange several low-dimensional rule base in a hierarchical structure, i.e., a tree, causing the number of possible rules to grow in a linear way according to the number of inputs. A method was proposed to determine automatically the fuzzy rules in a hierarchical fuzzy model [38]. Rainer [16] described a new algorithm which derives the rules for hierarchical fuzzy associative memories that were structured as a binary tree. Wang and Wei [12], [13], [19] proposed specific hierarchical fuzzy systems and its universal approximation property was proved. The approximation capabilities of hierarchical fuzzy systems was further analyzed by Zeng and Keane [52]. But the main problem lies in fact that this is a specific hierarchical fuzzy system which lacks flexibility in structure adaptation, and it is difficult to arrange the input variables for each sub-model. Lin and Lee [20] proposed a genetic algorithm based approach to optimize the hierarchical structure and the parameters of five-inputs hierarchical fuzzy controller for the low-speed control problem. Based on the analysis of importance of each input variable and the coupling between any two input variables, the problem of how to distribute the input variables to different (levels of) relational modules for incremental and aggregated hierarchical fuzzy relational systems was addressed [33].

Building a hierarchical fuzzy system is a difficult task. This is because we need to define the architecture of the system (the modules, the input variables of each module, and the interactions between modules), as well as the rules of each modules. Two approaches could be used to tackle this problem. One approach is that an expert supplies all the required knowledge for building the system. The other one is to use machine and/or optimization techniques to construct/adapt the system. Several machine learning and optimization techniques have been applied to aid the process of building hierarchical fuzzy systems. For example, Shimojima *et al.* use genetic algorithm to determine the hierarchical structure [38]. This is combined with backpropagation and gradient descent algorithm to fine tune its parameters. A structure identification method of sub-models for hierarchical fuzzy modeling using the multiple objective genetic algorithm was proposed by Tachibana and Furuhashi [39]. Chen

Manuscript received December 7, 2004 ; revised October 24, 2005 and February 2, 2006. This work was supported in part by the Natural Science Foundation of China under Contract 60573065, and by The Provincial Science and Technology Development Program of Shandong under Contract SDSP2004-0720-03.

Y. Chen and L. Peng are with the School of Information Science and Engineering, Jinan University, Jinan 250022, P. R. China (e-mail: yhchen@ujn.edu.cn).

B. Yang is with the State Key Laboratory of Advanced Technology for Materials Synthesis and Processing, Wuhan University of Science and Technology, Wuhan 430023, China (e-mail: yangbo@ujn.edu.cn).

A. Abraham is with the IITA Professorship Program, School of Computer Science and Engineering, Chung-Ang University, Seoul 156 756, South Korea (e-mail: ajith.abraham@ieee.org).

Digital Object Identifier 10.1109/TFUZZ.2006.882472

et al. [40] have proposed a hybrid method to optimize the hierarchical Takagi–Sugeno (TS) fuzzy model by using ant programming and particle swarm optimization algorithm.

From now onwards, the hierarchical structure or structure for short means the way of arrangement of hierarchical TS fuzzy systems and the position/selection of each input variable in the sub-fuzzy systems. The free parameters to be optimized including all the parameters used in the hierarchical TS fuzzy systems including membership function parameters for each fuzzy sets, the free parameters in the consequent parts of the fuzzy rule base for each sub-fuzzy systems.

This paper presents a systematic design method for the hierarchical TS-FS model. The hierarchical structure is evolved using a probabilistic incremental program evolution (PIPE) [21]–[23] with specific instructions, an algorithm originally used for automatic program synthesis. The fine tuning of the rule's parameters encoded in the structure is accomplished using evolutionary programming (EP). The proposed method interleaves both PIPE and EP optimizations. Starting with random structures and rules' parameters, it first tries to improve the hierarchical structure and then as soon as an improved structure is found, it fine tunes its rules' parameters. It then goes back to improve the structure again and, provided it finds a better structure, it again fine tunes the rules' parameters. This loop continues until a satisfactory solution (hierarchical TS-FS model) is found or a time limit is reached. The novelty of this paper is in the usage of evolutionary mechanism for selecting the important features and for constructing a hierarchical TS fuzzy model automatically.

The rest of the paper is organized as follows. A new encoding method and a computational model for the hierarchical TS-FS are given in Section II. An automatic design method for the hierarchical TS-FS is presented in Section III. Some simulation results and discussions related to system identification and time-series prediction problems are provided in Sections IV and V, respectively. Finally in Section VI we present some conclusions and future works.

TS Fuzzy Inference System (TS-FS)

Fuzzy inference systems are composed of a set of *IF–THEN* rules. A TS fuzzy model has the following form of fuzzy rules [1]:

$$\begin{aligned} R_j : \text{if } x_1 \text{ is } A_{1j} \text{ and } x_2 \text{ is } A_{2j} \text{ and } \ldots \\ \text{and } x_n \text{ is } A_{nj} \\ \text{Then } y = g_j(x_1, x_2, \ldots, x_n), \qquad (j = 1, 2, \ldots, N) \end{aligned}$$

where $g_j(\cdot)$ is a crisp function of x_i. Usually, $g_j(x_1, x_2, \ldots, x_n) = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_n x_n$. The overall output of the fuzzy model can be obtained by:

$$y = \frac{\sum_{j=1}^{N} g_j(\cdot) T_{i=1}^{m_j} \mu_{ij}(x_i)}{\sum_{j=1}^{N} T_{i=1}^{m_j} \mu_{ij}(x_i)} \tag{1}$$

where $1 \le m_j \le n$ is the number of input variables that appear in the rule premise, N is the number of fuzzy rules, n is the number of inputs, μ_{ij} is the membership function for fuzzy set, A_{ij} and T is a T-norm for fuzzy conjunction.

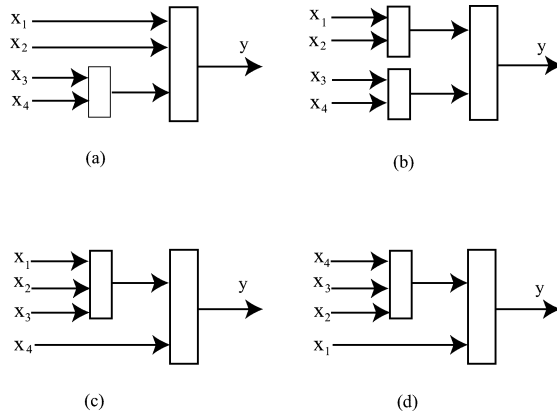

Fig. 1. Example of possible hierarchical fuzzy logic models, number of inputs: 4, number of layers: 3.

The TS-FS is a single-stage fuzzy system. It is important to partition the input space using some clustering, grid partitioning etc. [34]. The shapes of membership functions in the antecedent parts, and the free parameters in the consequent parts are also to be determined using some adaptive techniques [35], [36], [51].

II. Hierarchical TS-FS: Encoding and Evaluation

An hierarchical fuzzy inference system not only provides a more complex and flexible architecture for modelling nonlinear systems, but can also reduce the size of rule base to some extend. Fig. 1 depicts some possible hierarchical TS-FS models for 4 input variables and 3 hierarchical layers. The problems in designing a hierarchical fuzzy logic system include the following:

- selecting an appropriate hierarchical structure;
- selecting the inputs for each fuzzy TS sub-model;
- eetermining the rule base for each fuzzy TS sub-model;
- optimizing the parameters in the antecedent parts and the linear weights in the consequent parts.

There is no direct/systematic method for designing the hierarchical TS-FS. From the evolution point of view, finding a proper hierarchical TS-FS model can be posed as a search problem in the structure and parameter space. For this purpose, a new encoding method for hierarchical TS-FS is developed in Section II-A.

A. Encoding

A tree-structural based encoding method with specific instruction set is selected for representing a hierarchical TS-FS in this research. The reasons for choosing this representation are that: 1) the tree has a natural and typical hierarchical layer; and 2) with predefined instruction sets, the tree can be created and evolved using the existing tree-structure-based approaches, i.e., genetic programming (GP) and PIPE algorithms.

Assume that the used instruction set is $I = \{+_2, +_3, x_1, x_2, x_3, x_4\}$, where $+_2$ and $+_3$ denote nonleaf nodes' instructions taking 2 and 3 arguments, respectively. x_1, x_2, x_3, x_4 are leaf nodes' instructions taking zero arguments each. In addition, the output of each non-leaf

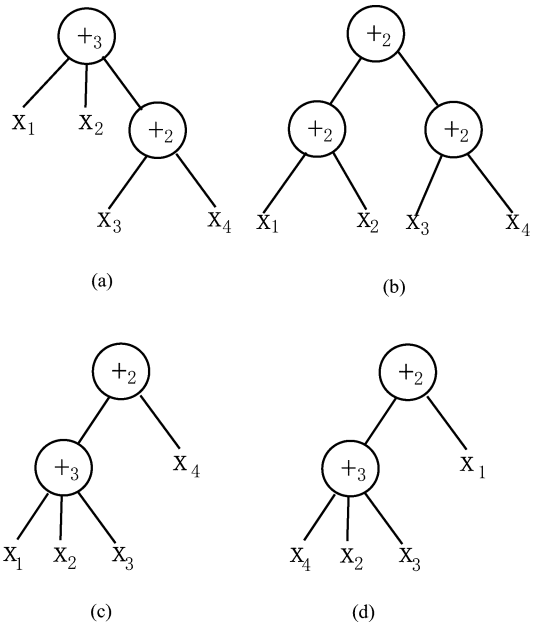

Fig. 2. Tree structural representation of the hierarchical T-S fuzzy models as shown in Fig. 1(a)–(d), where the used instruction set is $I = \{+_2, +_3, x_1, x_2, x_3, x_4\}$.

node is calculated as a single TS fuzzy sub-model. For this reason the non-leaf node $+_2$ is also called a two-input TS fuzzy instruction/operator. Fig. 2 illustrates the tree structural representation of the hierarchical TS fuzzy models (as per in Fig. 1).

It should be noted that in order to calculate the output of each TS fuzzy sub-model (non-leaf node), parameters in the antecedent parts and consequent parts of the TS fuzzy sub-model should be embedded into the tree.

B. Evaluation

In this subsection, we describe an illustrative example to show how the hierarchical TS-FS tree is calculated. The output of a hierarchical TS-FS tree can be calculated from a layer to layer. For simplicity, the calculation process of the tree [Fig. 2(a)] is illustrated later.

Assume that each input variable is divided into two fuzzy sets and the used fuzzy membership function is

$$\mu(a,b;x) = \frac{1}{1+\left(\frac{x-a}{b}\right)^2}. \tag{2}$$

First, the output of the TS fuzzy sub-model (node $+2$) is computed. Assume that the used fuzzy sets for variables x_3 and x_4 are A_{11}, A_{12} and A_{21}, A_{22}, respectively. Suppose that the parameters in the consequent parts of rule base are c_{ij}^0, c_{ij}^1, c_{ij}^2, ($i = 1, 2$ and $j = 1, 2$). These free parameters are encoded in the node $+_2$. Therefore, the corresponding fuzzy rules of node $+2$ can be described as:

$$\begin{aligned} &R_{i,j}: \text{if } x_3 \text{ is } A_{1i} \text{ and } x_4 \text{ is } A_{2j} \text{ then} \\ &y_{ij} = c_{ij}^0 + c_{ij}^1 x_3 + c_{ij}^2 x_4, \text{ for } i = 1,2 \text{ and } j = 1,2. \end{aligned}$$

The output of node $+_2$ can be calculated based on the TS fuzzy model

$$y = \frac{\sum_{i=1}^{2}\sum_{j=1}^{2} \sigma_{ij} y_{ij}}{\sum_{i=1}^{2}\sum_{j=1}^{2} \sigma_{ij}} \tag{3}$$

where

$$\sigma_{ij} = \mu_{A_{1i}}(x_3)\mu_{A_{2j}}(x_4) \text{ for } i = 1,2 \text{ and } j = 1,2.$$

Second, the overall output of the hierarchical TS fuzzy model is computed. It has three inputs, x_1, x_2 and y, the output of the TS fuzzy sub-model (node $+_2$). Assume that the used fuzzy sets for variables x_1, x_2 and y are: B_{11}, B_{12}, B_{21}, B_{22}, B_{31}, and B_{32}, respectively. Suppose that the parameters in the consequent parts of rule base are d_{ijl}^0, d_{ijl}^1, d_{ijl}^2, and d_{ijl}^3 ($i = 1, 2$, $j = 1, 2$, $l = 1, 2$). These free parameters are encoded in node $+_3$. The complete fuzzy rules of node $+_3$ can be described as follows:

$$\begin{aligned} &R_{i,j,l}: \text{if } x_1 \text{ is } B_{1i},\ x_2 \text{ is } B_{2j},\ y \text{ is } B_{3l} \text{ then} \\ &z_{ijl} = d_{ijl}^0 + d_{ijl}^1 x_1 + d_{ijl}^2 x_2 + d_{ijl}^3 y, \\ &\text{for } i = 1,2, \text{ and } j = 1,2 \text{ and } l = 1,2. \end{aligned}$$

Thus, the overall output of the tree is

$$z = \frac{\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{l=1}^{2} \mu_{ijl} z_{ijl}}{\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{l=1}^{2} \mu_{ijl}} \tag{4}$$

where

$$\mu_{ijl}(x_1, x_2, y) = \mu_{B_{1i}}(x_1)\mu_{B_{2j}}(x_2)\mu_{B_{3l}}(y).$$

It should be noted that all the parameters encoded in the tree are randomly generated along with the creation of the tree initially, which will be further optimized using evolutionary programming.

C. Objective Function

The fitness function used for the PIPE and EP is given by mean square error (MSE)

$$Fit_1(i) = \frac{1}{P}\sum_{j=1}^{P}\left(y_1^j - y_2^j\right)^2 \tag{5}$$

or root mean square error (RMSE)

$$Fit_2(i) = \sqrt{\frac{1}{P}\sum_{j=1}^{P}\left(y_1^j - y_2^j\right)^2} \tag{6}$$

where P is the total number of samples, y_1^j and y_2^j are the actual and hierarchical TS-FS model outputs of j -th sample. $Fit_1(i)$ and $Fit_2(i)$ denote the fitness value of the ith individual.

III. Evolutionary Design of Hierarchical TS-FS

In this section, an automatic design method of hierarchical TS-FS is presented. The hierarchical structure is created and optimized using PIPE with specific instructions and the fine turning of the rule's parameters encoded in the structure is accomplished using EP algorithm.

A. PIPE

PIPE [21]–[23] combines probability vector coding of program instructions, population based incremental learning [24], [25], and tree-coded programs [32]. PIPE iteratively generates successive populations of functional programs according to an adaptive probability distribution, represented as a probabilistic prototype tree (PPT), over all possible programs. Each iteration uses the best program to refine the distribution. Thus, the structures of promising individuals are learned and encoded in PPT.

1) Instructions and Programs: In PIPE, programs are made of instructions from an instruction set $S = \{I_1, I_2, \ldots, I_n\}$ with n instructions. Instructions are user-defined and problem dependent. Each instruction is either a function or a terminal. Instruction set S, therefore, consists of a function set $F = \{f_1, f_2, \ldots, f_k\}$ with k functions and a terminal set $T = \{t_1, t_2, \ldots, t_l\}$ with l terminals, where $n = k + l$ holds.

Programs are encoded in n -ary trees, with n being the maximal number of function arguments. Each nonleaf node encodes a function from F and each leaf node a terminal from T. The number of subtrees each node has corresponds to the number of arguments of its function. Each argument is calculated by a subtree. The trees are parsed depth first from left to right.

2) PPT: The PPT stores the knowledge gained from experiences with programs and guides the evolutionary search. It holds random constants and the probability distribution over all possible programs that can be constructed from a predefined instruction set. The PPT is generally a complete n-ary tree with infinitely many nodes, where n is the maximal number of function arguments.

Each node N_j in PPT, with $j > 0$ contains a random constant R_j and a variable probability vector $\vec{P}_j$. Each $\vec{P}_j$ has n components, where n is the number of instructions in instruction set S. Each component $P_j(I)$ of $\vec{P}_j$ denotes the probability of choosing instruction $I \in S$ at node N_j. Each vector $\vec{P}_j$ is initialized as follows:

$$P_j(I) = \frac{P_T}{l} \qquad \forall\, I : I \in T \tag{7}$$

$$P_j(I) = \frac{1 - P_T}{k} \qquad \forall\, I : I \in F \tag{8}$$

where l is the total number of terminals in T, k is the total number of functions in F, and P_T is initially user-defined constant probability for selecting an instruction from T.

3) Program Generation, Growing and Pruning: Programs are generated according to the probability distribution stored in the PPT. To generate a program P_{ROG} from PPT, an instruction $I \in S$ is selected with probability $P_j(I)$ for each accessed node N_j of PPT. Nodes are accessed in a depth-first way, starting at the root node and traversing PPT from left to right.

A complete PPT is infinite, and each PPT node holds a probability for each instruction, a random constant, and n pointers to following nodes, where n is PPT's arity. Therefore, a large PPT is memory intensive. To reduce memory requirements, it is thus possible to incrementally grow and prune the PPT.

On one hand, it is useful to grow the PPT on demand in order to create a variety of programs. Initially the PPT contains only the root node. Additional nodes are created with each program that accesses nonexisting nodes during its generation. On the other hand, apart from reducing memory requirements, pruning also helps to discard the elements of probability distribution that have become irrelevant over time. PPT subtrees attached to nodes that contain at least one probability vector component above a threshold T_P can be pruned. If T_P is set to a sufficiently high value (e.g., $T_P = 0.99999$) only parts of the PPT will be pruned that have a very low probability of being accessed. In case of functions, only those subtrees should be pruned that are not required as function arguments. Fig. 3 illustrates the relation between the prototype tree and a possible program tree.

4) Fitness Functions: Similar to the other evolutionary algorithms, PIPE uses a problem-dependent and user-defied fitness function. A fitness function maps programs to scalar, real-valued fitness values that reflect the programs' performances on a given task. Firstly PIPE's fitness functions should be seen as error measures, i.e., MSE [(5)] or RMSE [(6)]. A secondary nonuser-defined objective for which PIPE always optimizes programs is the program size as measured by number of nodes. Among programs with equal fitness smaller ones are always preferred. This objective constitutes PIPE's built-in Occam's razor.

5) Learning Algorithm: PIPE combines two forms of learning: Generation-based learning (GBL) and elitist learning (EL). GBL is PIPE's main learning algorithm. EL's purpose is to make the best program found so far as an attractor. PIPE executes:

GBL

REPEAT

with probability P_{el} DO EL

otherwise DO GBL

UNTIL termination criterion is reached.

Here, P_{el} is a user-defined constant in the interval $[0, 1]$.

Generation-Based Learning

Step 1. **Creation of Program Population.** A population of programs $P_{ROG\,j}$ ($0 < j \leq PS$; PS is population size) is generated using the prototype tree PPT, as described in **Section III-A** . The PPT is grown on demand.

Step 2. **Population Evaluation.** Each program P_{ROG_j} of the current population is evaluated on the given task and assigned a fitness value $FIT(P_{ROG_j})$ according to the predefined fitness

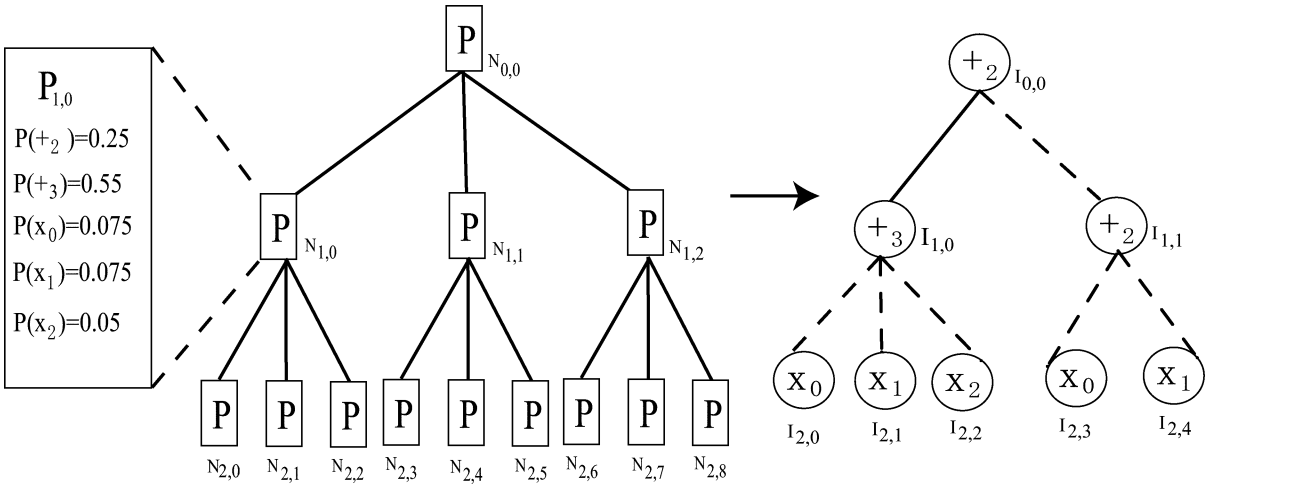

Fig. 3. Example of node $N_{1,0}$'s instruction probability vector $P_{1,0}$ (left). Probabilistic prototype tree PPT (middle). Possible extracted program P_{ROG}, at the time of creation of instruction $I_{1,0}$, the dashed part of P_{ROG} did not exist yet (right).

function. The best program of the current population (the one with the smallest fitness value) is denoted P_{ROG_b}. The best program found so far (elitist) is preserved in ${P_{ROG}}^{el}$.

Step 3. **Learning from Population.** Prototype tree probabilities are modified such that the probability $P(P_{ROG_b})$ of creating P_{ROG_b} increases. This procedure is called adapt PPT towards(Progb). This is implemented as follows. First $P(P_{ROG_b})$ is computed by looking at all PPT nodes N_j used to generate P_{ROG_b}:

$$P\left(P_{ROG_b}\right) = \prod_{j:N_j \text{ used to generate } P_{ROG_b}} P_j\left(I_j(P_{ROG_b})\right) \tag{9}$$

where $I_j(P_{ROG_b})$ denotes the instruction of program P_{ROG_b} at node position j. Then a target probability P_{TARGET} for P_{ROG_b} is calculated:

$$P_{\text{TARGET}} = P(P_{ROG_b}) + (1 - P(P_{ROG_b})) \cdot lr \cdot \frac{\varepsilon + FIT\left({P_{ROG}}^{el}\right)}{\varepsilon + FIT(P_{ROG_b})} \tag{10}$$

here lr is a constant learning rate and ε a positive user-defined constant. Given P_{TARGET}, all single node probabilities $P_j(I_j(P_{ROG_b}))$ are increased iteratively.

REPEAT:

$$P_j\left(I_j(P_{ROG_b})\right) = P_j\left(I_j(P_{ROG_b})\right) + c^{lr} \cdot lr \cdot \left(1 - P_j\left(I_j(P_{ROG_b})\right)\right). \tag{11}$$

UNTIL $P\left(P_{ROG_b}\right) \geq P_{\text{TARGET}}$

where c^{lr} is a constant influencing the number of iterations. The smaller c^{lr} the higher the approximation precision of P_{TARGET} and the number of required iterations. Setting $c^{lr} = 0.1$ turned out to be a good compromise between precision and speed. Then, all adapted vectors $\vec{P}_j$ are renormalized.

Step 4. **Mutation of Prototype Tree.** All probabilities $P_j(I)$ stored in nodes N_j that were accessed to generate program P_{ROG_b} are mutated with a probability P_{M_p}:

$$P_{M_p} = \frac{P_M}{n \cdot \sqrt{|P_{ROG_b}|}} \tag{12}$$

where the user-defined parameter P_M defines the overall mutation probability, n is the number of instructions in instruction set S and $|P_{ROG_b}|$ denotes the number of nodes in program P_{ROG_b}. Selected probability vector components are then mutated as follows:

$$P_j(I) = P_j(I) + mr \cdot (1 - P_j(I)) \tag{13}$$

where mr is the mutation rate, another user-defined parameter. Also all mutated vectors $\vec{P}_j$ are renormalized.

Step 5. **Prototype Tree Pruning.** At the end of each generation the prototype tree is pruned, as described in **Section III-B** .

Step 6. **Termination Criteria.** Repeat above procedure until a fixed number of program evaluations is reached or a satisfactory solution is found.

Elitist Learning

Elitist learning focuses search on previously discovered promising parts of the search space. The PPT is adapted towards the elitist program ${P_{ROG}}^{el}$. This is realized by replacing the P_{ROG_b} with ${P_{ROG}}^{el}$ in learning from population in Step 3. It is particularly useful with small population sizes and works efficiently in the case of noise-free problems.

B. Special Instructions for Evolving the Hierarchical TS-FS

Evolving structured program is a fundamental approach for restricting search space and speeding up evolution both in GP and PIPE areas. The basic PIPE algorithm has been extended by using hierarchical instructions and skip nodes in hierarchical PIPE (H-PIPE) [22], [23].

However, H-PIPE cannot be directly used for evolving the hierarchical TS-FS due to its linear representation of hierarchical layers. To create an appropriate hierarchical TS-FS architecture, a specified instruction set is proposed in this research

$$I = \{+_2, +_3, \ldots, +_N, x_1, x_2, \ldots, x_n\} \tag{14}$$

where $+_i (i = 2, 3, \ldots, N)$ is i-inputs TS fuzzy instruction as discussed at Sections II-A and B. $x_1, x_2, \ldots, x_n$ are inputs variables. and n is the number of the inputs variables. Nonterminal instruction, $+_i (i = 2, 3, \ldots, N)$ plays a key role for generating the hierarchical layer (stage) of the TS-FS. The input selection method on how to assign inputs to different hierarchical layer is accomplished by the PIPE algorithm automatically.

It should be noted that the hierarchical TS-FS tree may grows large and contains redundant information without any constrains on the hierarchical layer (maximum depth in hierarchical TS-FS tree). Therefore, for implementing the algorithm, the maximum number of hierarchical layers is predefined in our experiments.

C. Parameter Optimization

1) Basics of Evolutionary Programming: Evolutionary algorithms constitute a universal optimization method that imitates the type of genetic adaptation that occurs in natural evolution [37]. Unlike specialized methods designed for particular types of optimization tasks, they require no particular knowledge about the problem structure other than the objective function itself. A population of candidate solutions evolves over time by means of genetic operators such as mutation, recombination and selection. The different types of evolutionary algorithms, namely genetic algorithms (GAs), evolution strategies (ES), evolutionary programming (EP), and genetic programming (GP) all share the same principle mode of operation, but utilize different genetic representations and operators.

Compared to GAs, the primary search operator in EP is mutation. One of the major advantages of using mutation-based EA is that they can reduce the negative impact of the permutation problem. Hence the evolutionary process can be more efficient. Gaussian mutation has been the most commonly used mutation operator, but Cauchy mutation and other mutation operators can also be used [29], [30]. Evolving free parameters of the hierarchical TS-FS model by EP can be implemented as follows.

i) Generate the initial population of μ individuals, and set iteration step $k = 1$. Each individual is taken as a pair of real-valued vectors, $(x_i, \eta_i), \forall i \in \{1, \ldots, \mu\}$, where x_i's are free parameter vectors encoded in the hierarchical TS-FS tree and η_i's are variance vectors for Gaussian mutations (also known as strategy parameters in self-adaptive EAs). Each individual corresponds to a hierarchical TS-FS tree.
ii) Evaluate the fitness score [(5) or (6)] for each individual of the population according to a task at hand.
iii) Each parent (x_i, η_i), $i = 1, \ldots, \mu$, creates a single offspring $(\bar{x}_i, \bar{\eta}_i)$ by

$$\bar{\eta}_i(j) = \eta_i(j) \exp(\bar{\tau} N(0,1) + \tau N_j(0,1)) \tag{15}$$

$$\bar{x}_i(j) = x_i(j) + \bar{\eta}_i(j) \cdot N_j(0,1) \tag{16}$$

for $j = 1, \ldots, n$, where $x_i(j)$, $\bar{x}_i(j)$, $\eta_i(j)$ and $\bar{\eta}_i(j)$ denote the jth component of the vectors x_i, $\bar{x}_i$, η_i and $\bar{\eta}_i$, respectively. $N(0,1)$ denotes a normally distributed random number. $N_j(0,1)$ indicates that the random number is generated anew for each value of j; and $\tau = (\sqrt{2\sqrt{n}})^{-1}$, $\bar{\tau} = (\sqrt{2n})^{-1}$ [27], [28].
iv) Calculate the fitness of each offspring.
v) Conduct pairwise comparison over union of parents and offspring. For each individual, Q opponents are chosen randomly from all the parents and offspring with an equal probability. For each comparison, if the individual's fitness is not smaller than the opponents', the individual wins.
vi) Select the μ individual out of parents and offspring, that has the most wins to qualify as a parent of the next generation.
vii) Stop if the number of EP search iterations have reached; otherwise, $k = k + 1$ and go to step iii).

2) Enhancements of the Basic EP: The convergence speed of EP algorithm depends largely on the search steps. In order to control the convergence speed and enhance the performance of EP search, we introduced a scale factor α, and a dynamic factor $\sigma(k)$ shown in the (17) into the conventional EP

$$\sigma(k) = \alpha \cdot \left(1 - 0.9 \cdot \frac{k}{K}\right). \tag{17}$$

The scale factor α is used to control step size of the search, the variable term $(1 - 0.9 \cdot k/K)$ is used for further tuning of the search precision, where k is the current generation number varying from 0 to K, K is the maximum generation number of EP search. Therefore, the update (16) is replaced by

$$\bar{x}_i(j) = x_i(j) + \sigma(k) \cdot \bar{\eta}_i(j) \cdot N_j(0,1). \tag{18}$$

D. The Proposed Algorithm for Designing of Hierarchical TS-FS Model

Combining the self-organizing and structure learning characteristics of PIPE and the parameter optimization ability of EP, we propose the following hybrid algorithm for designing the hierarchical TS-FS model (Fig. 4).

1) Set the initial values of parameters used in the PIPE and EP algorithms. Set the elitist program as NULL and its fitness value as a biggest positive real number of the computer at hand. Create the initial population (tree) and corresponding parameters used in hierarchical TS-FS model.
2) Do structure optimization using PIPE algorithm as described in Section III-A, in which the fitness function is calculated by (5) or (6).
3) If the better structure found, then go to step 4), otherwise go to step 2). The criterion concerning with better structure found is distinguished as follows: If the fitness value of the best program is smaller than the fitness value of the elitist program, or the fitness values of two programs are equal but the nodes of the former is lower than the later, then we say that the better structure is found.

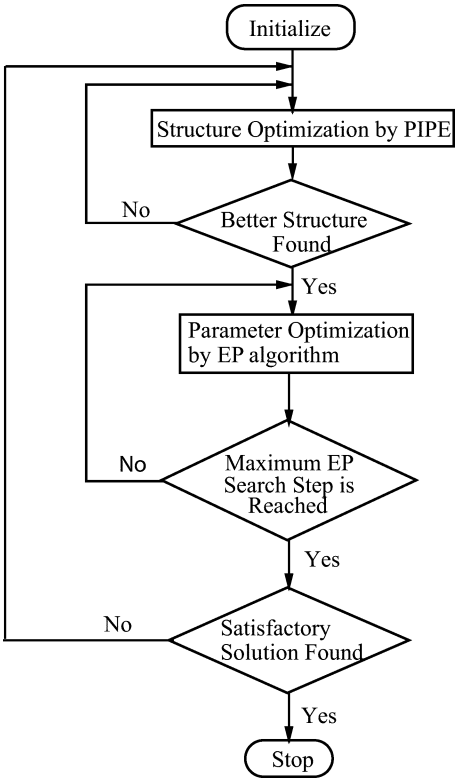

Fig. 4. Flow chart of the proposed algorithm for designing the hierarchical TS-FS model.

4) Parameter optimization using EP search as described in Section III-C. In this step, the tree structure or architecture of hierarchical TS-FS model is fixed, and it is the best tree taken from the end of run of PIPE search. All of the rules' parameters encoded in the best tree will be optimized by EP search in order to decrease the fitness value of best program.
5) If the maximum number of EP search is reached, or no better parameter vector is found for a significantly long time (100 steps) then go to step 6); otherwise go to step 4).
6) If satisfactory solution is found, then stop; otherwise go to step 2).

E. Feature/Input Selection With Hierarchical TS-FS

It is often a difficult task to select important variables for prediction and classification problems, especially when the feature space is large. A predefined single/multilevel fuzzy model usually cannot do this. In the perspective of evolution-driven hierarchical TS-FS framework, the nature of model construction procedure allows the H-TS-FS to identify important input features in building an prediction or classification model that is computationally efficient and effective. The mechanisms of input selection in the H-TS-FS constructing procedure are as follows.

- Initially the input variables are selected to formulate the Hierarchical TS-FS model with same probabilities.
- The variables that has more contribution to the objective function will be enhanced and have high opportunity to survive in the next generation by a evolutionary procedure.

TABLE I
PARAMETERS USED IN THE PIPE ALGORITHM

Parameters	Values
population size PS	100
elitist learning probability P_{el}	0.01
learning rate lr	0.01
fitness constant ε	0.000001
overall mutation probability P_M	0.4
mutation rate mr	0.4

- The evolutionary operators, i.e., crossover and mutation, provide a input selection method by which the hierarchical TS-FS would select the appropriate variables automatically.

IV. EXPERIMENTS

The proposed approach has been evaluated for nonlinear system identification problems, Mackey–Glass chaotic time-series prediction problem, and the Iris and Wine classification problems. Sections IV-A–D discuss these applications and the results obtained by the evolutionary design of hierarchical TS-FS model and the performance is compared with other fuzzy/neural learning approaches.

The used parameters in PIPE are shown in Table I. The parameters used in EP: population size is 60, opponent number $Q = 30$, $\alpha = 0.3$. For all the simulations, the minimum and maximum number of hierarchical layers are predefined as 2 and 4 and each input variable is partitioned into 2 fuzzy sets. The used fuzzy membership function is shown in (2). The initial fuzzy rules for each sub-fuzzy systems are randomly generated and all the free parameters including fuzzy sets membership function parameters and the free parameters in the consequent parts of fuzzy rules are randomly generated at $[0, 1]$ initially. It should be noted that the selection of the nonleaf's instruction is experimental. Selecting more instructions will increase the structure/parameter search space and results in a bigger hierarchical TS fuzzy system. For an identification or classification problem, if the input number is n, selecting the maximum instruction $+_N$ as $N = n/3$ is enough according to our experiments. This experimental rule should reduce the search space significantly.

A. Systems Identification

The first plant to be identified is a linear system given by [26]

$$\begin{aligned} y(k+1) = {} & 2.627771y(k) - 2.333261y(k-1) \\ & + 0.697676y(k-2) + 0.017203u(k) \\ & - 0.030862u(k-1) \\ & + 0.014086u(k-2). \end{aligned} \tag{19}$$

400 data points were generated with the randomly selected input signal $u(k)$ between -1.0 and 1.0. The first 200 points were used as training data set and the remaining data were used as validation data set. The input vector is set as $x = [y(k), y(k-1), y(k-2), u(k), u(k-1), u(k-2)]$. The used instruction set is $I = \{+_2, +_3, +_4, x_0, x_1, x_2, x_3, x_4, x_5\}$.

10 independent runs were taken. The average training time for ten runs is 245 seconds. The best structure of evolved hierarchical TS-FS model is shown in Fig. 5(a). The output of the

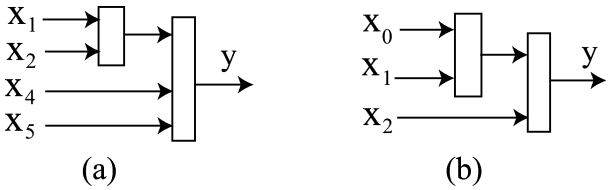

Fig. 5. Structure of evolved hierarchical TS-FS models. (a) Plant 1. (b) Plant 2.

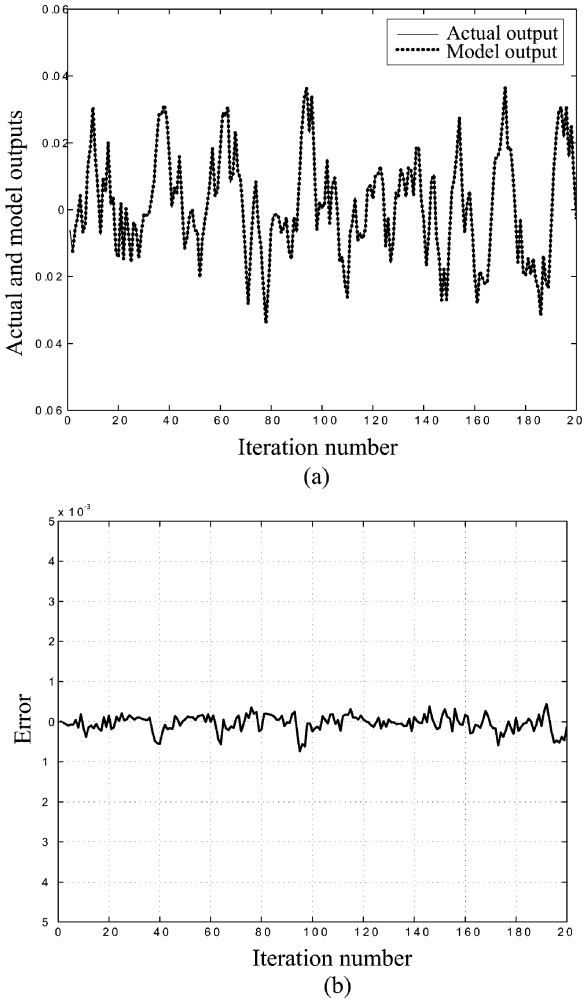

Fig. 6. Actual and evolved outputs of the plant 1 for (a) the test data set and (b) the test error.

evolved model, the actual output and the test error for test data set are illustrated in Fig. 6.

The second plant to be identified is a nonlinear system given by [26]

$$y(k+1) = \frac{y(k)}{1.5 + y^2(k)} - 0.3y(k-1) + 0.5u(k). \quad (20)$$

The input and output of system are $x(k) = [u(k), u(k-1), y(k), y(k-1)]$ and $y(k+1)$, respectively.

The training samples and the test data set are generated by using the same sequence of random input signals as mentioned previously. The used instruction set is $I = \{+_2, +_3, +_4, x_0, x_1, x_2, x_3\}$.

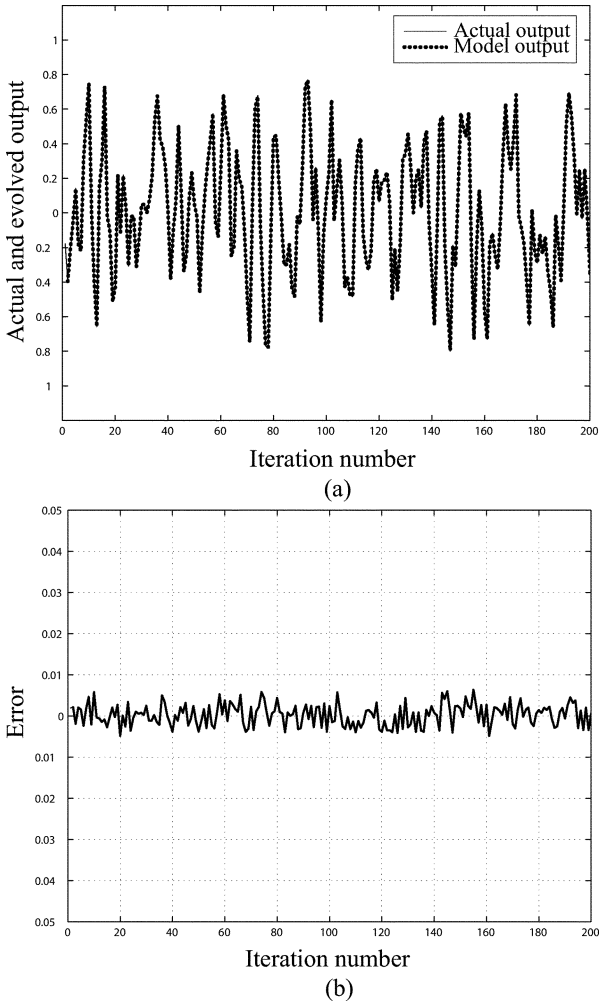

Fig. 7. Actual and evolved outputs of the plant 2 for (a) the test data set and (b) the test error.

TABLE II
THE COMPARISON OF THE MSE VALUES FOR MODIFIED ELMAN NETS [26], MODIFIED JORDAN NETS [26], WAVELET NEURAL NETWORKS (WNN) [41] AND HIERARCHICAL TS-FS MODEL FOR TEST DATA SET

Plant	Elman	Jordan	WNN	H-TS-FS
1	0.0000548	0.0000492	0.000000459	0.0000000432
2	0.0004936	0.0003812	0.000002728	0.0000007065

Ten independent runs were run. The average training time of ten runs is 317 seconds. The best structure of evolved hierarchical TS-FS model is shown in Fig. 5(b). The output of the evolved model, the actual output and the test error for test data set are shown in Fig. 7.

For comparison, the test results obtained by Elman and Jordan neural networks [26], Wavelet Neural Networks (WNN) [41] and the proposed H-TS-FS model are shown in Table II. From the above simulation results, it is evident that the proposed hierarchical TS-FS model works very well for identifying the linear/nonlinear systems much better than the neural network models.

B. Chaotic Time-Series of Mackey–Glass

The Mackey–Glass chaotic differential delay equation is recognized as a benchmark problem that has been used

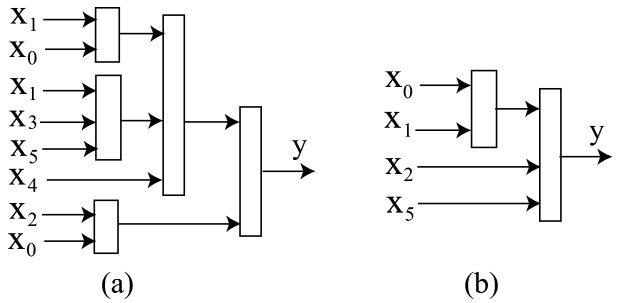

Fig. 8. Two possible structures of hierarchical TS-FS models for predicting the Mackey–Glass time-series: With (a) $\mathrm{RMSE} = 0.01205$ and (b) $\mathrm{RMSE} = 0.01417$.

and reported by a number of researchers for comparing the learning and generalization ability of different models. The Mackey–Glass chaotic time series is generated USING the following differential equation:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x^{10}(t-\tau)} - bx(t) \tag{21}$$

where $a = 0.2$ and $b = 0.1$, $\tau > 17$ the equation shows chaotic behavior. In our simulations, $\tau = 30$ has been adopted. To compare with previous works [33], we predicted the value of $x(t+6)$ using the input variables $x(t-30), x(t-24), x(t-18), x(t-12)$, $x(t-6)$ and $x(t)$, where $t = 130$ to $t = 1129$. It corresponds to a 6-input to 1-output mapping.

1000 sample points were used in our study. The first 500 data pairs were used as training data, while the remaining 500 were used to validate the model identified. The used instruction set is $I = \{+_2, +_3, x_0, x_1, x_2, x_3, x_4, x_5\}$, where $x_0, x_1, x_2, x_3, x_4, x_5$ denote $x(t-30)$, $x(t-24)$, $x(t-18)$, $x(t-12)$, $x(t-6)$, and $x(t)$, respectively.

The results are obtained from training the hierarchical TS-FS models using 10 different experiments. The average training time of ten runs is 719 s. The average RMSE value for training and test data sets are 0.017 and 0.015, respectively.

Two evolved structures of hierarchical TS-FS models are shown in Fig. 8. A comparison has been made to show the actual time-series, the hierarchical TS-FS model output and the prediction error [Fig. 9(a)]. Fig. 9(b) shows the convergence performance of the best hierarchical TS-FS model. Performance comparison of the different methods for approximating the Mackey–Glass data is shown in Table III.

C. Iris Data Classification

The Iris data is a common benchmark in classification and pattern recognition research [42]. It contains 50 measurements of four features from each of the three species Iris setosa, Iris versicolor, and Iris virginica [44]. We label the species 1–3, respectively, which gives a 5×150 pattern matrix of observation vectors

$$Z_k^T = \left[x_1^k, x_2^k, x_3^k, x_4^k, c_k\right], \quad c_k \in 1,2,3, \quad k = 1,2,\ldots,150 \tag{22}$$

where x_1^k, x_2^k, x_3^k, and x_4^k are the sepal length, sepal width, petal length, and petal width, respectively.

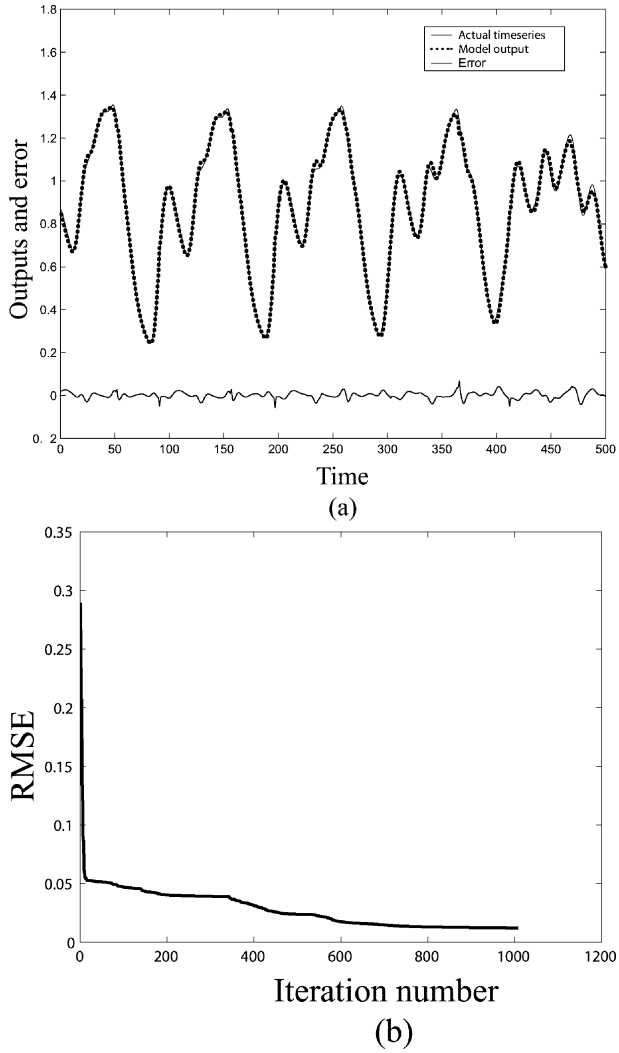

Fig. 9. Actual time-series, model output and prediction error for (a) test data set and (b) fitness curve for training.

TABLE III
COMPARISON OF THE INCREMENTAL TYPE MULTILEVEL FRS (IFRS) [33], THE AGGREGATED TYPE MUTILEVEL FRS (AFRS) [33], AND THE HIERARCHICAL TS-FS IN MACKEY -GLASS TIME -SERIES PREDICTION (H-TS-FS1 AND H-TS-FS2 ARE CORRESPONDING TO THE MODEL STRUCTURES SHOWN IN FIG. 8 (RIGHT) AND FIG. 8 (LEFT), RESPECTIVELY)

Model	Stage	Rules	Parameters	RMSE training	RMSE testing
IFRS	4	25	58	0.0240	0.0253
AFRS	5	36	78	0.0267	0.0256
H-TS-FS1	3	28	148	0.0120	0.0129
H-TS-FS2	2	12	46	0.0145	0.0151

In our simulations, we normalized each attribute value into a real number in the unit interval. Table IV shows the results of some well-known classifier systems. For the Iris example, we also used 150 patterns to design a hierarchical TS-FS classifier system via the proposed algorithm. The used instruction set is $F = \{+_2, +_3, x_1, x_2, x_3, x_4\}$.

Table V shows the results of ten runs (i.e., ten different initializations of parameters). To estimate the performance of the proposed method on unseen data, the five-fold cross-validation was

TABLE IV
COMPARISON OF RESULTS FOR IRIS DATA

	Term sets	Rules	Recognition rate (%)
Wang et al. [45]	11	3	97.5
Wu et al. [46]	9	3	96.2
Shi et al. [5]	12	4	98.0
Russo [47]	18	5	100
Ishibuchi et al. [43]	7	5	98.0
This paper	-	16	99.6

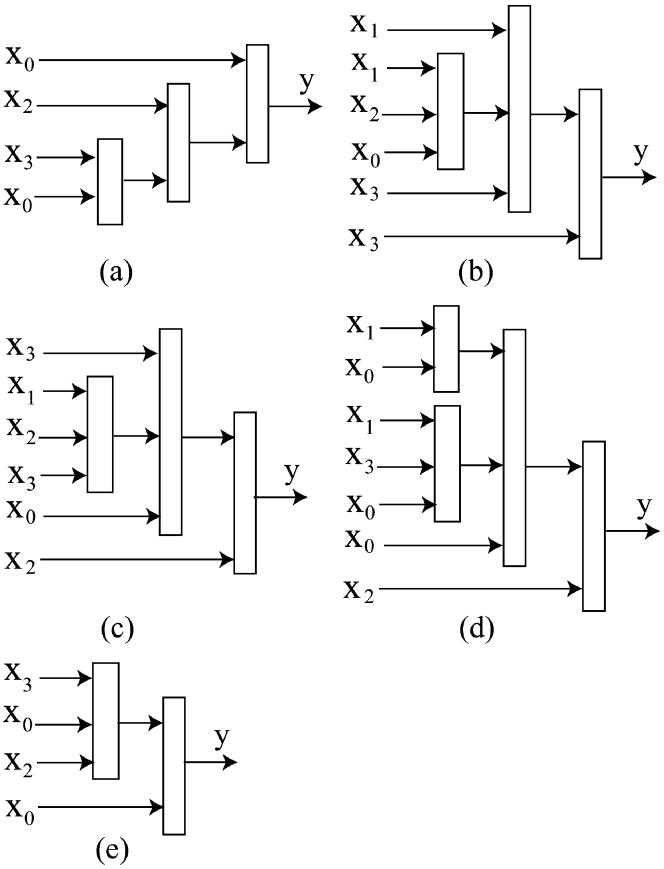

Fig. 10. Evolved optimal H-TS-FS architectures for five-fold cross-validation (Iris data).

performed on the iris data. In the five-fold cross-validation experiment, the normalized iris data were divided into five disjoint groups containing 30 different patterns each, with ten patterns belonging to each class. Then we derived the hierarchical TS-FS models via the proposed method on all data outside one group and tested the resulting hierarchical TS-FS classifier on the data within that group. Finally, five hierarchical TS-FS were derived. The evolved hierarchical architectures for five-fold cross-validation are shown in Fig. 10. The convergence performance of five-fold cross validation test 3 is shown in Fig. 11(a). Table VI reports the results of five-fold cross validation. The average classification result is 100.0% correct (no misclassifications) on the training data and 99.34% correct (average about 0.2 misclassification) on the test data using 17.6 rules (average).

D. Wine Data Classification

The proposed hierarchical TS-FS model is applied to wine data. The wine data set is a 13-dimensional problem with 178 samples from three classes. We chose this data set because it involves many continuous attributes. We normalized each attribute value into a real number in the unit interval.

Table VII shows the results of some well-known classifier systems. For the wine data, we also used 178 patterns to design a

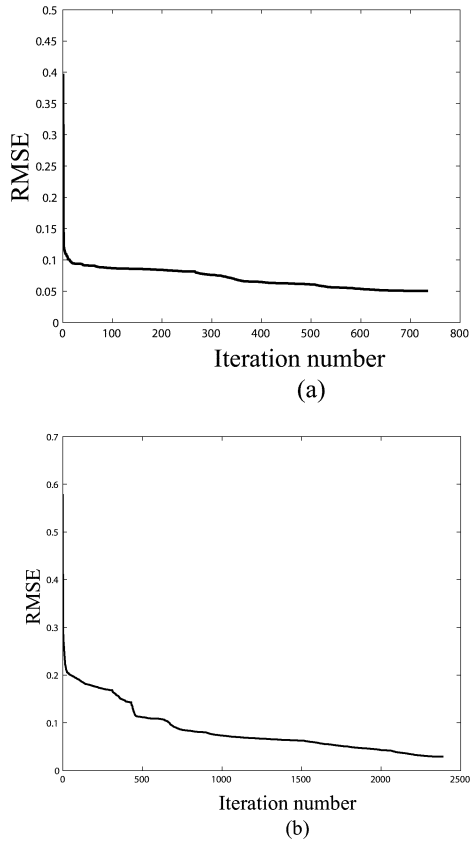

Fig. 11. Convergence performance of five-fold cross validation test 3 for (a) Iris data and (b) Wine data.

TABLE V
RESULTS OF TEN RUNS ON IRIS DATA

	1	2	3	4	5	6
Misclassification	1	1	0	0	0	1
Recognition rate (%)	99.3	99.3	100	100	100	99.3
Features	4	4	3	4	4	3
Rules	12	12	16	20	20	12
Parameters	64	60	84	108	108	60
Training time (m)	8	12	21	17	22	9
	7	8	9	10	Average	
Misclassification	1	1	1	0	0.6	
Recognition rate (%)	99.3	99.3	99.3	100	99.6	
Features	4	4	3	4	3.7	
Rules	12	16	20	20	16	
Parameters	64	84	104	108	84.4	
Training time (m)	19	12	25	11	16.7	

TABLE VI
FIVE-FOLD CROSS VALIDATION FOR IRIS DATA

	1	2	3	4	5	Aver.(%)
Rules	12	20	20	24	12	17.6
Training patterns	120	120	120	120	120	120
Misclassification (train)	0	0	0	0	0	0
Recog. rate (train)(%)	100	100	100	100	100	100
Testing patterns	30	30	30	30	30	30
Misclassification (test)	0	0	0	0	1	0.2
Recog. rate (test)(%)	100	100	100	100	96.7	99.34

TABLE VII
COMPARISON OF RESULTS USING WINE DATA

	Features	Term sets	Rules	Recog. rate(%)
Setnes et al. [48]	9	21	3	98.3
Wang et al. [45]	13	34	3	99.4
Roubos et al. [49]	5	15,11,10	3	98.9, 98.3. 99.4
Ishibuchi et al. [43]	-	9	6	100
This paper	4.9	-	16.4	99.6

TABLE VIII
RESULTS OF TEN RUNS ON WINE DATA

	1	2	3	4	5	6
Misclassification	0	1	1	1	1	0
Recognition rate (%)	100	99.4	99.4	99.4	99.4	100
Features	5	4	4	5	5	6
Rules	16	12	12	20	16	20
Parameters	84	60	64	108	84	108
Training time (minutes)	10	14	23	19	24	13
	7	8	9	10	Average	
Misclassification	1	1	0	1	0.7	
Recognition rate (%)	99.4	99.4	100	99.4	99.6	
Features	4	6	6	4	4.9	
Rules	12	20	20	16	16.4	
Parameters	64	108	108	84	87.2	
Training time (minutes)	22	24	28	18	19.5	

hierarchical TS-FS classifier system via the proposed algorithm. The used instruction set is $F = \{+_2, +_3, +_4, x_1, x_2, \ldots, x_{13}\}$.

Table VIII illustrates the empirical results of ten runs (i.e., ten different initializations of parameters). To estimate the performance of the proposed method on unseen data, the five-fold cross-validation was performed on the Wine data. In the five-fold cross-validation experiment, the normalized Wine data were divided into five disjoint groups. Then we derived the hierarchical TS-FS models via the proposed method on all data outside one group and tested the resulting hierarchical TS-FS classifier for the data within that group.

Finally, five hierarchical TS-FS were derived. The evolved hierarchical architectures for five-fold cross-validation are shown in Fig. 12. The convergence performance of five-fold cross validation test 3 is shown in Fig. 11(b). Table IX reports the results of five-fold cross validation. The average classification result is 100.0% correct (no misclassifications) on the training data and 99.4% correct (average about 0.2 misclassification) on the test data using 22.4 rules (average).

V. DISCUSSIONS

One major advantage of using a hierarchical TS-FS or a mutilevel fuzzy system other than a single-level system (direct approach) is that the number of fuzzy rules and fuzzy operations involved in modeling process can be reduced significantly when compared with those required by the single-level counterparts. Due to the limitations to solve the hierarchical TS-FS analytically, we choose to identify the hierarchical TS-FS using an evolutionary optimization approach.

First, the hierarchical structure and the rules' parameters can be flexibly encoded into a TS-FS tree. And then, the PIPE and the EP algorithms are employed to evolve the optimal structure and parameters alternatively. The methods used by IFRS and AFRS [33], the hierarchical structure and input selection are assigned based on: 1) analysis of the importance of each input variables; and 2) analysis of the coupling between input variables. In contrast to the IFRS and AFRS, the hierarchical structure and input selection in this research are accomplished using an evolutionary procedure automatically.

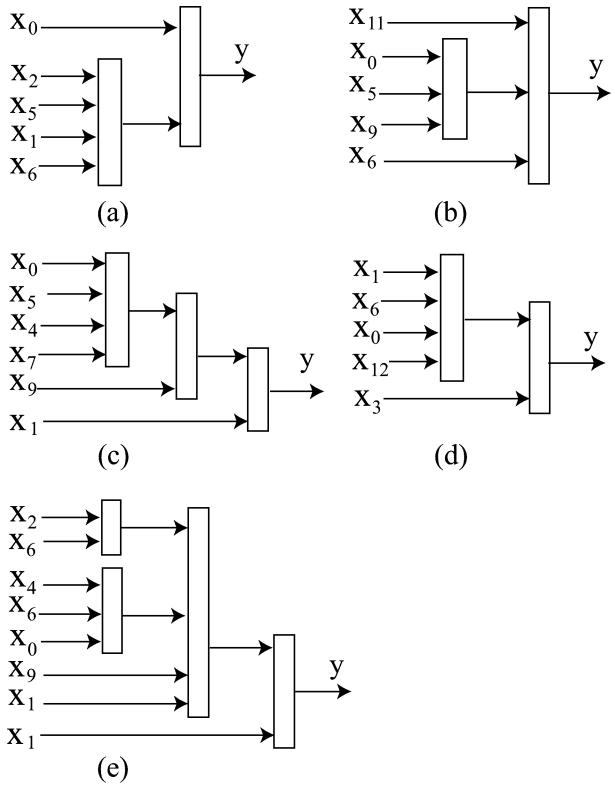

Fig. 12. Evolved optimal H-TS-FS architectures for five-fold cross-validation (Wine data).

TABLE IX
FIVE -FOLD CROSS VALIDATION FOR WINE DATA

	1	2	3	4	5	Aver.(%)
Rules	20	16	24	20	32	22.4
Training patterns	136	144	144	144	144	142.4
Misclassification (train)	0	0	0	0	0	0
Recog. rate (train)(%)	100	100	100	100	100	100
Testing patterns	42	34	34	34	34	35.6
Misclassification (test)	0	1	0	0	0	0.2
Recog. rate (test)(%)	100	97.1	100	100	100	99.4

The effectiveness of the proposed methods has been demonstrated through various benchmark problems. Furthermore, compared to the IFRS and AFRS the generated hierarchical TS-FS model has some advantages in terms of the approximation accuracy, the number of rules and the number of free parameters.

This paper provides a method that alternatively searches between the tree-structure space and parameter space by using the PIPE and EP algorithms. But other tree-structure based evolutionary algorithms and parameter learning algorithms can also be used to solve the problem.

VI. CONCLUSION

Based on a novel representation and computational model of the hierarchical TS-FS model, an approach for evolving the hierarchical TS-FS was proposed in this paper. The hierarchical architecture and inputs selection method of the hierarchical TS-FS

were accomplished using PIPE algorithm, and the rules' parameters embedded in the hierarchical TS-FS model were optimized using a enhanced EP algorithm. Simulation results shown that the evolved hierarchical TS-FS models are effective for the identification of linear/nonlinear systems, for the prediction of chaotic time-series, and for the classification of Iris and Wine data.

It should be noted that the hierarchical TS-FS has smaller number of rules than a single level (direct approach) TS-FS. The number of rules and parameters would increase tremendously (even difficult to manage) for large number of inputs if a direct approach is used. This also results in slow convergent speed. Our future works will concentrate on the following.

- Embedding rules' deduction techniques in the proposed hierarchical TS-FS.
- Improving the convergence speed of the proposed method by parallel implementation of the algorithm.
- Apply the proposed approach to more complex problems (real world applications).

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for the technical suggestions and remarks which helped to improve the contents and the quality of this presentation.

REFERENCES

[1] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst. Man, Cybern.*, vol. SMC-15, no. 1, pp. 116–132, Jan. 1985.

[2] C. Xu and Y. Liu, "Fuzzy model identification and self learning for dynamic systems," *IEEE Trans. Syst. Man, Cybern.*, vol. SMC-17, pp. 683–689, Jul./Aug. 1987.

[3] V. Torra, "A review of the construction of hierarchical fuzzy systems," *Int. J. Intell. Syst.*, vol. 17, pp. 531–543, 2002.

[4] Q. Gan and C. J. Harris, "Fuzzy local linearization and logic basis function expension in nonlinear system modeling," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 4, pp. 559–565, Aug. 1999.

[5] Y. Shi, R. Eberhart, and Y. chen, "Implementation of evolutionary fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 2, pp. 109–119, Apr. 1999.

[6] C. -K. Lin and S.-D. Wang, "Fuzzy system identification using an adaptive learning rule with terminal attractors," *J. Fuzzy Sets Syst.*, pp. 343–352, 1999.

[7] S. -J. Kang, C. -H. Woo, H. -S. Hwang, and K. B. Woo, "Evolutionary design of fuzzy rule base for nonlinear system modeling and control," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 1, pp. 37–45, Feb. 2000.

[8] Y. -P. Huang and S. -F. Wang, "Designing a fuzzy model by adaptive macroevolution genetic algorithms," *Fuzzy Sets Syst.*, vol. 113, pp. 367–379, 2000.

[9] B. Wu and X. Yu, "Fuzzy modelling and identification with genetic algorithm based learning," *Fuzzy Sets Syst.*, vol. 113, pp. 352–365, 2000.

[10] M. Denna, G. Mauri, and A. M. Zanaboni, "Learning fuzzy rules with tabu search-an application to control," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 2, pp. 295–318, Apr. 1999.

[11] G. V. Raju and J. Zhou, "Adaptive hierarchical fuzzy controller," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 4, pp. 973–980, Aug. 1993.

[12] L. X. Wang, "Analysis and design of hierarchical fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 5, pp. 617–624, Oct. 1999.

[13] ——, "Universal approximation by hierarchical fuzzy systems," *Fuzzy Sets Syst.*, vol. 93, pp. 223–230, 1998.

[14] O. Huwendiek and W. Brockmann, "Function approximation with decomposed fuzzy systems," *Fuzzy Sets Syst.*, vol. 101, pp. 273–286, 1999.

[15] K. Hiroaki *et al.*, "Functional completeness of hierarchical fuzzy modeling," *Inf. Sci.*, vol. 110, no. 1–2, pp. 51–60, 1998.

[16] H. Rainer, "Rule generation for hierarchical fuzzy systems," in *Proc. Annu.Conf. North American Fuzzy Information Processing*, 1997, pp. 444–449.

[17] K. Sun-Yuan *et al.*, "Synergistic modeling and applications of hierarchical fuzzy neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1550–1574, Sep. 1999.

[18] M. Brown, K. M. Bossley, D. J. Mills, and C. J. Harris, "High dimensional neurofuzzy systems: Overcoming the curse of dimensionality," in *Proc. 4th Int. Conf. Fuzzy Systems*, 1995, pp. 2139–2146.

[19] C. Wei and L. -X. Wang, "A note on universal approximation by hierarchical fuzzy systems," *Inf. Sci.*, vol. 123, pp. 241–248, 2000.

[20] L. C. Lin and G. -Y. Lee, "Hierarchical fuzzy control for C-axis of CNC tuning centers using genetic algorithms," *J. Intell. Robot. Syst.*, vol. 25, no. 3, pp. 255–275, 1999.

[21] R. P. Salustowicz and J. Schmidhuber, "Probabilistic incremental program evolution," *Evol. Comput.*, vol. 2, no. 5, pp. 123–141, 1997.

[22] ——, "Evolving structured programs with hierarchical instructions and skip nodes," in *Machine Learning: Proc. 15th Int. Conf. (ICML98)*, San Francisco, CA, 1998, pp. 488–496.

[23] J. Schmidhuber, On learning how to learn learning strategies Fakultaet fuer Informatik, Technische Universitaet, Muenchen, Germany, 1995, Tech. Rep. FKI-198–94.

[24] M. A. Wiering and J. Schmidhuber, L. Saitta, Ed., "Solving POMDPs with levin search and EIRA," in *Machine Learning: Proc. 13th International Conference*, San Francisco, CA, 1996, pp. 534–542.

[25] S. Baluja, Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning Carnegie Mellon Univ., Pittsburgh, PA, 1994, Tech. Rep. CMU-CS-94–163.

[26] D. T. Pham and D. Karaboga, "Training elman and jordan networks for system identification using genetic algorithms," *Artif. Intell. Eng.*, vol. 13, pp. 107–117, 1999.

[27] D. B. Fogel, K. Chellapilla, and P. J. Angeline, "Inductive reasoning and bounded rationality reconsidered," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 142–146, Jul. 1999.

[28] D. B. Fogel, E. C. Wasson, and E. M. Boughton, "Evolving neural networks for detecting breast cancer," *Cancer Lett.*, vol. 96, pp. 49–53, 1995.

[29] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep. 1999.

[30] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[31] H. Ying, "Theory and application of a novel fuzzy PID controller using a simplified Takagi-Sugeno rule scheme," *Inf. Sci.*, vol. 123, no. 3–4, pp. 281–293, 2000.

[32] N. L. Cramer, J. Grefenstette, Ed., "A representation for the adaptive generation of simple sequential programs," in *Proc. Int. Conf. Genetic Algorithms Appl.*, Hillsdale, NJ, 1985, pp. 183–187.

[33] J. -C. Duan and F. -L. Chung, "Multilevel fuzzy relational systems: Structure and identification," *Soft Comput.*, vol. 6, pp. 71–86, 2002.

[34] R. Babuska, "Fuzzy modeling and identification," Ph.D. dissertation, Univ. Delft, Delft, The Netherlands, 1996.

[35] P. Angelov and D. Filev, "An approach to online identification of Takagi–Sugeno fuzzy models," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 484–498, Feb. 2004.

[36] N. Kasabov and Q. Song, "DENFIS: Dynamic, evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 1, pp. 144–154, Feb. 2002.

[37] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 2nd ed. Piscataway, NJ: IEEE Press, 1999.

[38] K. Shimojima, T. Fukuda, and Y. Hasegawa, "Self-turning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm," *Fuzzy Sets Syst.*, vol. 71, pp. 295–309, 1995.

[39] K. Tachibana and T. Furuhashi, "A structure identification method of submodels for hierarchical fuzzy modeling using the multiple objective genetic algorithm," *Int. J. Intell. Syst.*, vol. 17, pp. 495–531, 2002.

[40] Y. Chen, B. Yang, and J. Dong, "Automatic design of hierarchical TS-FS models using ant programming and PSO algorithm," in *Proc. 11th Int. Conf. Artificial Intelligence: Methodology, Systems, Applications*, 2004, vol. LNCS 3192, pp. 285–294.

[41] Y. Chen and S. Kawaji, "Evolving wavelet neural networks for system identification," in *Proc. Int. Conf. Electrical Engineering (ICEE2000)*, Kitakyushu, Japan, 2000, pp. 279–282.

[42] H. Ishibuchi, T. Murata, and I. B. Turksen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets Syst.*, vol. 89, pp. 135–150, 1997.

[43] H. Ishibuchi, T. Nakashima, and T. Murata, "Three-objective geneticbased machine learning for linguistic rule extraction," *Inf. Sci.*, vol. 136, pp. 109–133, 2001.
[44] E. Anderson, "The irises of the gaspe peninsula," *Bull. Amer. Iris Soc.*, vol. 59, pp. 2–5, 1935.
[45] J. S. Wang and G. C. S. Lee, "Self-adaptive neuro-fuzzy inference system for classification application," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 6, pp. 790–802, Dec. 2002.
[46] T. P. Wu and S. M. Chen, "A new method for constructing membership functions and fuzzy rules from training examples," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 1, pp. 25–40, Feb. 1999.
[47] M. Russo, "Genetic fuzzy learning," *IEEE Trans. Evol. Comput.*, vol. 4, no. 5, pp. 259–273, Sep. 2000.
[48] M. Setnes and H. Roubos, "GA-fuzzy modeling and classification: Complexity and performance," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 509–522, Oct. 2000.
[49] J. A. Roubos, M. Setnes, and J. Abonyi, "Learning fuzzy classification rules from labeled data," *Inf. Sci.*, vol. 150, no. 1–2, pp. 77–93, Mar. 2003.
[50] A. Abraham, *Adaptation of Fuzzy Inference System Using Neural Learning, Fuzzy System Engineering: Theory and Practice*, N. Nedjah, Ed. *et al.* Berlin, Germany: Springer-Verlag, 2005, ch. 3, pp. 53–83.
[51] ——, "EvoNF: A framework for optimization of fuzzy inference systems using neural network learning and evolutionary computation," in *Proc. 17th IEEE Int. Symp. Intelligent Control*, 2002, pp. 327–332.
[52] X. -J. Zeng and J. A. Keane, "Approximation capabilities of hierarchical fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 5, pp. 659–672, Sep. 2005.
[53] M. G. Joo and J. S. Lee, "A class of hierarchical fuzzy systems with constraints on the fuzzy rules," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 2, pp. 194–203, Apr. 2005.
[54] S. Paulo, "Clustering and hierarchization of fuzzy systems," *Soft Comput. J.*, vol. 9, no. 10, pp. 715–731, 2005.

Yuehui Chen (M'02) was born in 1964 in Shandong Province, China. He received the B.Sc. degree in mathematics/automatics from Shandong University, China, in 1985, and the M.S. and Ph.D. degree in electrical engineering from the Kumamoto University, Japan, in 1999 and 2001, respectively.

During 2001–2003, he had worked as the Senior Researcher of the Memory-Tech Corporation, Tokyo, Japan. Since 2003, he has been a member of the Faculty of Electrical Engineering, Jinan University, Jinan, China, where he is currently Head of the Laboratory of Computational Intelligence. His research interests include evolutionary computation, neural networks, fuzzy logic systems, hybrid computational intelligence and their applications in time-series prediction, system identification, intelligent control, intrusion detection systems, web intelligence and bioinformatics. He is the author and coauthor of more than 70 technique papers.

Prof. Chen is a member of the IEEE Systems, Man and Cybernetics Society and the Computational Intelligence Society, a member of Young Researchers Committee of the World Federation on Soft Computing, and a member of the CCF Young Computer Science and Engineering Forum of China. He is Editor-in-Chief of the *Journal of Computational Intelligence in Bioinformatics* and the Associate Editor of the *International Journal of Computational Intellignece Research*. More information can be found at http://cilab.ujn.edu.cn.

Bo Yang received the B.Eng. degree from Jinan University, Jinan, China, in 1984, and the M.S. degree from South China University, in 1989.

He is a Professor and Vice-President at Jinan University, Jinan, China. He is the Director of the Provincial Key Laboratory of Information and Control Engineering, and also acts as the Associate Director of Shandong Computer Federation, and Member of the Technical Committee of Intelligent Control of Chinese Association of Automation. His main research interests include computer networks, artificial intelligence, machine learning, knowledge discovery, and data mining. He has published numerous papers and gotten some important scientific awards in this area.

Ajith Abraham (M'96) received the Ph.D degree from Monash University, Melbourne, Australia, in 2001.

He currently works as a Professor under the South Korean Government's Institute of Information Technology Assessment (IITA) Professorship program at Chung-Ang University, Korea. He is also a Visiting Researcher at Rovira i Virgili University, Spain, and an Adjunct Professor at Jinan University, Jinan, China and Dalian Maritime University, Dalian, China. His primary research interests are in computational intelligence with a focus on using evolutionary computation techniques for designing intelligent paradigms. Application areas include Web services, information security, Web intelligence, financial modeling, multicriteria decision-making, data mining, etc. He has authored/coauthored over 200 research publications in peer-reviewed reputed journals, book chapters and conference proceedings, three of which have won Best Paper awards. He is serving on the Editorial Boards of over a dozen international journals and has also guest edited 15 special issues for reputed international journals. Since 2001, he has been actively involved in the Hybrid Intelligent Systems (HIS) and the Intelligent Systems Design and Applications (ISDA) series of annual international conferences. More information can be found at http://www.softcomputing.net.

Lizhi Peng received the B.S. and M.E. degrees in computer science from Xi'an Jiaotong University and Jinan University, China, in 1998 and 2004, respectively.

He is currently a Research Assistant in the Department of Information Science and Engineering, Jinan University. His main research interests include computational intelligence, data mining, and bioinformatics. He has published numerous papers in these areas.