An Evolutionary Fuzzy Modeling Approach for ANFIS Architecture

Farzad Rastegar[†], Babak N. Araabi[†‡], and Caro Lucas[†‡]
[†]Control and Intelligent Processing Center of Excellence, Electrical and Computer Eng. Department, University of Tehran, Tehran, Iran
and
[‡]School of Intelligent Systems, Institute for studies in theoretical Physics and Mathematics, Tehran, Iran
f.rastegar@ece.ut.ac.ir, araabi@ut.ac.ir, lucas@ipm.ir

Abstract- This paper proposes a cooperative evolutionary method for optimizing the properties of an ANFIS-architecture-based model where only the input-output data of the identified system are available. The primary tasks of fuzzy modeling are structure identification and parameter optimization: the former determines the numbers of membership functions and fuzzy if-then rules while the latter identifies a feasible set of parameters under the given structure. The proposed approach manages all mentioned attributes simultaneously. Particularly, number of rules and parameters of membership functions are realized by applying a novel approach using genetic programming and genetic algorithm whereas consequent parameters are tuned by using least-squares estimation. Finally, two examples of nonlinear system are given to illustrate the effective-ness of the proposed approach.

1 Introduction

Fuzzy modeling has been studied to deal with complex, ill-defined and uncertain systems, in which the conventional mathematical models fail to give satisfactory results [25]. The main purpose of fuzzy modeling is to describe the behavior of a system by a set of fuzzy inference rules. In the literature there has been a vast array of various approaches for the modeling [6], [16], [18], [21]. One of the most influential fuzzy models has proposed by Jang in [6] called ANFIS (Adaptive-Network-Based Fuzzy Inference System). The rule base of this model contains the fuzzy if-then rules of Takagi and Sugeno's type in which consequent parts are linear functions of the inputs instead of fuzzy sets [11], reducing the number of required fuzzy rules.

The identification of a fuzzy model consists of two major phases: structure identification and parameter optimization [2], [4], [7], [14], [20], [23], [24]. The first phase is the determination of the number of fuzzy if-then rules and the membership functions of the premise fuzzy sets while the second phase is the tuning of the parameter values of the fuzzy model. In comparison with parameter optimization, structure identification is a more difficult task, and often is tackled by off-line trial-and-error approaches, like evolutionary algorithms [3], [5], [15].

Evolutionary algorithms are optimization methods that mimic the processes in natural evolution and genetics. Genetic algorithms (GA), evolutionary strategies (ES), evolutionary programming (EP) and genetic programming (GP) are the prominent approaches. Genetic programming proposed by Koza [10] is concerned with the automatic generation of computer programs by means of simulated evolution. GP has been applied to a remarkable variety of different domains, such as symbolic regression, electronic circuit design, data mining, biochemistry, robot control, optimization, pattern recognition, planning and evolving game-playing strategies.

In this study we propose a novel cooperative evolutionary approach along with least-squares to obtain the number of rules, parameters of membership functions and consequent parameters of fuzzy rules in ANFIS architecture in order to achieve a neuro-fuzzy system which can accurately model nonlinear systems from given input-output data. First, we exploit two sets of training and test data pairs from the identified system. Second, a special architecture of ANFIS is designed to be used as a framework to test the candidate solutions. Third, various solutions are developed through a cooperative evolutionary method applying genetic programming and genetic algorithm, and then they are tested by the framework designed in the previous step to gradually reach the accurate solution which satisfies the error tolerance.

The cooperative evolutionary approach along with the least-squares helps to develop an accurate model. It is worth stressing that there have been a number of interesting approaches toward evolutionary development of fuzzy models [3], [5], [8], [9], [16]. This study differs from them in the following distinct ways.

- The majority of suggested evolutionary methods in the literature treat only one or two sub-tasks separately. In most cases, the input partition is predetermined or the structure and parameter identification are performed in a segregated way. It should be noted that the learning sub-tasks are related to each other. In that respect a separate manipulation of them may lead to sub-optimal solutions. For example, the determination of the proper number of rules is highly related to the size, the location and the overlapping of the fuzzy regions in a partition. In this paper we propose an approach in which two evolutionary algorithms cooperatively attempt to optimize the structure and parameters of membership functions all together.

0-7803-9363-5/05/$20.00 ©2005 IEEE.

- The existence of a mechanism designed in this approach to keep the optimized parameter sets of various structures over generations makes it possible that the new solutions whose structures have been met in past generations can inherit the evolved parameter sets of their families - members which have the same structure are called a family - and continue the evolution to reach more optimized parameter sets (see section 4).
- This cooperative evolutionary approach proposes an implicit competition beside the explicit evolution for various candidate structures which are produced in generations. Every family implicitly competes with each other in how fast it can evolve its parameters of membership functions to reach the lower error. The faster a family evolves its parameters, the more chance it has to survive. On the other hand, the speed of evolution is not considered with equal weight for all families. The more complex the family, the more speed it needs to survive. The criterion by which a family is labeled complex is the number of rules which it suggests.
- Since the proposed approach optimizes the attributes of a fuzzy model simultaneously, the convergence of this approach to an optimized solution occurs fast and the attained solution is noticeably accurate.

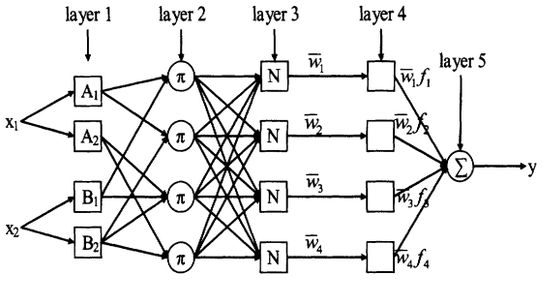
Experimental results illustrate that the proposed approach outperforms the prior works (see section 5) with regard to model simplicity and the training and testing error. This paper is organized in six sections. In section 2, we introduce the ANFIS architecture and the framework which is utilized in this paper. In section 3, the least-squares approach applied for optimization of consequent parameters is elaborated. The details of the cooperative evolutionary approach which is contributed in this paper are discussed in section 4. Simulation results of a non-linear system and the comparison with those reported in prior studies are given in section 5.

2 ANFIS

Jang's ANFIS [6] is a 5-layered feedforward neuro-fuzzy network whose node functions of the same layer are of the same function family. ANFIS applies the rules of TSK (Takagi, Sugeno and Kang) form in its architecture. Let us consider a physical system with m input variables $x = [x_1, x_2, ..., x_m]^T$ and a single output, y, with $x_i \in D_i = [x_{i,min}, x_{i,max}] \subseteq R$, $(i = 1, .., m)$ and $y \in Y = [y_{min}, y_{max}] \subseteq R$. The TSK fuzzy rules are of the following form:

$$R^{(r)}: \quad IF\ x_1\ is\ A^r_1\ AND\ ...\ AND\ x_m\ is\ A^r_m \quad THEN\ y_r = g^r(x) = \alpha^r_0 + \alpha^r_1 x_1 + ... + \alpha^r_m x_m \quad (r = 1, ..., n). \tag{1}$$

where n is the number of rules, x_i $(i=1, ..., m)$ are the model inputs, and y_r is the output of the rule $R^{(r)}$. A^r_i $(i = 1, ..., m)$ are fuzzy sets in D_i characterized by membership functions $\mu_{A^r_i}(x_i)$, and α^r_i are real coefficients of the rule polynomials, $g^r(x)$. the m-tuple $(A^r_1, ..., A^r_m)$ of fuzzy sets in the premise part of $R^{(r)}$ forms a fuzzy region $A^r_1 \times ... \times A^r_m$ in $D = D_1 \times ... \times D_m$, called the fuzzy hyper-cube (FHC) in this paper.



Layer	1	2	3	4	5
Node	L*V	R	R	R	1
Parameter	2	0	0	V+1	0

Fig. 1. An adaptive-network-based fuzzy inference system. The table at bottom shows the number of nodes and the number of parameters of each node at each layer. V is the number of input variables; L is the number of linguistic terms of each input; and R is the number of rules. In this case $R=L\times V$

Considering a first-order TSK fuzzy inference system which contains the following rules:
Rule r: If x_1 is A^1_i and x_2 is A^2_j, then $y_r = \alpha^r_0+\alpha^r_1 x_1+\alpha^r_2 x_2$
where $r \in \{1, ..., 4\}$ and $i, j \in \{1, 2\}$, the corresponding ANFIS architecture is depicted in Fig. 1. Circles in ANFIS represent fixed nodes while squares are the representatives for adaptive nodes. Fixed nodes function as predefined operators to their inputs and no other parameters but the inputs participate in their calculations. Adaptive nodes, on the other hand, have some internal parameters which affect the results of their calculations.

Layer 1 is an adaptive layer which denotes membership functions to each input. In this paper we choose Gaussian functions as membership functions:

$$O^1_i = \mu_{A_i}(x) = \exp\left(\frac{-(x-c)^2}{2\sigma^2}\right). \tag{2}$$

where x is the input to node i; A_i is the membership function associated with this node; and {c, σ} is the parameter set that changes the shapes of the membership function. Parameters in this layer are referred to as the *premise parameters*.

Layer 2 is a fixed layer in which each node calculates the firing strength of each rule via multiplication:

$$O^2_i = w_i = \mu_{A_i}(x_1) \times \mu_{B_i}(x_2), i = 1,2,3,4. \tag{3}$$

Layer 3 is a fixed layer in which the i-th node calculates the ratio of the i-th rule's firing strength to the sum of all rule's firing strength:

$$O^3_i = \overline{w}_i = \frac{w_i}{\sum_{j=1}^{4} w_j}, i = 1,2,3,4. \tag{4}$$

Layer 4 is an adaptive layer in which the i-th node deals with the consequent parameters of the i-th rule as mentioned in (1). Node i in this layer has the following node function:

$$O_i^4 = \overline{w}_i y_i = \overline{w}_i(\alpha_1^i x_i + \alpha_2^i x_2 + \alpha_3^i), i = 1,2,3,4. \quad (5)$$

where $\overline{w}_i$ is the output of i-th node of layer 3 and $\{\alpha^i_1, \alpha^i_2, \alpha^i_3\}$ is the parameter set. Parameters in this layer are referred to as the *consequent parameters*.

Layer 5 is a fixed single-node layer which computes the overall output as the summation of all incoming signals:

$$O_1^5 = overall\_output = \sum_{i=1}^{4} \overline{w}_i y_i = \frac{\sum_{i=1}^{4} w_i y_i}{\sum_{i=1}^{4} w_i} \quad (6)$$

From Fig. 1, it is easy to see that the topological complexity of Jang's general model is $O(R)$ or $O(L^V)$. This is because of the assumption that the feature space is uniformly partitioned along each input dimension. Since the size of the network grows exponentially as the number of inputs increases, obviously the operation efficiency will be poor when the architecture is used to model a complicated system with many variables. As we do not use grid partitioning [17] for structure identification, a special ANFIS architecture suggested in [20] is pursued which has the complexity of $O(R \times V)$ with R independent of V. Fig. 2 shows the ANFIS architecture used in this paper.

The number of membership functions for each input variable in the architecture shown in Fig. 2 is equal to the number of rules. In this topology each membership function participates in only one rule.

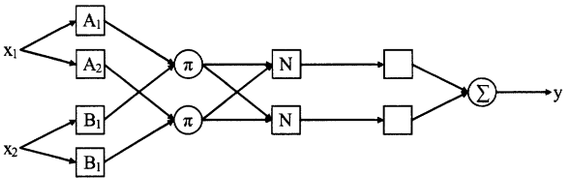
3 Optimization of Consequent Parameters

As equation (1) explains, the consequent part of a TSK fuzzy rule is a linear combination of input variables. However, equation (6) reveals that other elements, i.e. w_i's, are also involved in construction of the output of ANFIS architecture. The output of ANFIS, which is composed of input variables and firing strength of each rule, can be restated as follows:

$$\begin{aligned} y &= \overline{w}_1 y_1 + \overline{w}_2 y_2 + ... + \overline{w}_n y_n \\ &= \overline{w}_1 \alpha_0^1 + (\overline{w}_1 x_1)\alpha_1^1 + ... + (\overline{w}_1 x_m)\alpha_m^1 + ... \\ &\quad + \overline{w}_n \alpha_0^1 + (\overline{w}_n x_1)\alpha_1^n + ... + (\overline{w}_n x_m)\alpha_m^n \end{aligned} \quad (7)$$

where n and m are number of rules and number of input variables respectively.

If the parameters of membership functions and number of rules are identified, the equation (7) will become a linear equation in terms of consequent parameters. Therefore the consequent parameters can be efficiently estimated from training data. The three most common approaches are *global least squares* used in [6], [19], [21], *local weighted least squares* utilized in [5], [12], [13], and *product space clustering* [1], [20].



Layer	1	2	3	4	5
Node	R*V	R	R	R	1
Parameter	2	0	0	V+1	0

Fig. 2. The framework applied in cooperative evolution

In this paper the global least squares is pursued due to its advantages in terms of low error and high accuracy. With the help of equation (7), the parameter vector, X, regression matrix, A, and the output matrix, B, are defined as follow:

$$\begin{aligned} &AX = B, \\ &A = \begin{bmatrix} \overline{w}_1 & \overline{w}_1 x_1^1 & ... & \overline{w}_1 x_m^1 & ... & \overline{w}_n & \overline{w}_n x_1^1 & ... & \overline{w}_n x_m^1 \\ ... & & & & & & & & \\ \overline{w}_1 & \overline{w}_1 x_1^p & ... & \overline{w}_1 x_m^p & ... & \overline{w}_n & \overline{w}_n x_1^p & ... & \overline{w}_n x_m^p \end{bmatrix}, \\ &X = \begin{bmatrix} \alpha_0^1 & \alpha_1^1 & ... & \alpha_m^1 & ... & \alpha_0^n & \alpha_1^n & ... & \alpha_m^n \end{bmatrix}^T, \\ &B = \begin{bmatrix} y^{(1)} & y^{(2)} & y^{(3)} & ... & y^{(p)} \end{bmatrix}^T. \end{aligned} \quad (8)$$

The dimension of A and X are p×(n(m+1)) and (n(m+1))×1 respectively, where p is number of training data pairs, m is number of input variables and n is number of rules. Since the matrix A is not generally square, the pseudo-inverse of A is used:

$$X^* = (A^T A)^{-1} A^T B \quad (9)$$

As a result the optimized consequent parameters are attained in X^*.

4 The Cooperative Evolutionary Approach

This section illustrates the details of the approach applied in this paper to find the optimized fuzzy model based on ANFIS architecture.

4.1 Scenario

The purpose of this paper, as mentioned earlier, is to find the number of rules and parameters of membership functions for a fuzzy model so that the model has accurate output in comparison with that of identified system. To reach this goal, two evolutionary algorithms, i.e., GP and GA, cooperatively attempt to generate an optimized model inspiring this fact that obtaining an optimal solution

requires optimization of all involved factors together. Fig. 3 shows how the cooperation between GP and GA is performed.

First, GP initializes its first generation in which each chromosome specifies how the feature space should be partitioned. The chromosomes produced in each generation of GP make a pool of solutions for the structure identification of the fuzzy model (see section 4.2). Then, proper values are assigned to parameters (parameters of membership functions) of the structures proposed by the chromosomes according to whether or not their corresponding chromosomes have been met before. After that, every structure is sent to GA. GA evolves the structure's parameters to make it more accurate. Then, GA saves the optimized parameters of the structure into the database and sends it back with the evolved parameter set to GP. When all the chromosomes of the generation are performed as the same, GP produces the chromosomes of the next generation by using crossover and mutation operators. This process will continue until a structure with satisfying error is attained or a predefined number of generations are generated. The following pseudo-code explains the aforementioned process:

Cooperative evolution pseudo-code
g = 0
GP initializes generation$_0$
while the condition is not satisfied do
{
for i = 1..number of chromosomes in generation$_g$
{
chr$_i$ = the i-th chromosome in generation$_g$
if chr$_i$ is already met in past generations
GP retrieves the values of chr$_i$'s parameters from DB
else
GP assigns values to chr$_i$'s parameters according to the way chr$_i$ partitions the feature space
GA optimizes chr$_i$'s parameters and saves the optimized parameters into DB
}
GP generates next generation (generation$_{g+1}$) by means of crossover and mutation operators
g = g + 1
}
end

The cooperative evolution makes different families compete with one another to upraise their own fitness values (The word *family* refers to the chromosomes which partition the feature space in a same manner). In the mean time the members of a family cooperate with each other to optimize the parameters of their membership functions as much as they can. This simultaneous competition and cooperation makes this approach produce impressive results in comparison with the others. Moreover, the optimization imposed by GA on every GP's chromosome, causes faster convergence. Hence, a desirable result can be obtained by fewer generations.

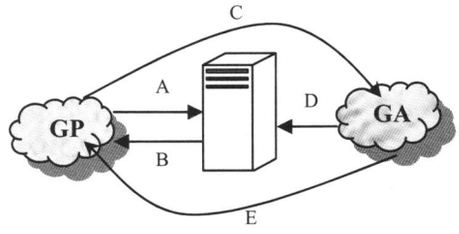


Fig. 3. Cooperative evolution. (A) GP retrieves the evolved parameters corresponding to the chromosomes of current generation from database, (B) the parameters are received by GP, (C) GP sends the chromosomes with the optimized parameters ever found to GA, (D) GA evolves the parameters further and saves them into database, (E) the chromosomes with more evolved parameters are sent back to GP.

4.2 Genetic Programming (GP)

GP is concerned with the automatic generation of computer programs. Most GP systems are represented as tree structures which have two kinds of nodes, function and terminal. Terminal nodes refer to nodes at the leaves of the tree and either provide a value, such as constant or correspond to primitive action such as a robot motion command. Function nodes, on the other hand, correspond to the non-leaf nodes in the tree and compute or process the information passed up from their children.

In this paper, GP is utilized as a Pittsburgh approach [17] for structure identification [5]. It deals with trees whose terminal set is $T = \{L\}$ and function set is $F = \{F_1, F_2, ..., F_m\}$ where m is the number of input variables. The first generation is seeded with variety of different trees generated randomly. A tree is generated by starting with the root node and recursively adding nodes to left and right of each node (binary tree). The type of each node is declared as a terminal or function randomly. The process of generating nodes for a tree will be terminated when every branch ends with a terminal node. If the length of a branch exceeds from a maximal depth, it will be forced to stop growing by a terminal node.

The selection method used by GP is the tournament selection, in which a group of K chromosomes (trees) are randomly selected. From these candidates only the best chromosome is selected for reproduction, whereas the others are discarded. The tournament is repeated until the population is filled with a new generation of offspring.
The Crossover operator exchanges a subtree taken from a parent with another subtree of the other parent. First crossover randomly selects a subtree from each parent. Subsequently, the subtrees are cut off and swapped between the parents resulting in to new offspring.

The mutation operator applied for GP in this paper is a point mutation. Point mutation can be used to change the symbols of terminal or function nodes. Our approach employs point mutation over function nodes. It randomly alters the function symbol of an inner node; as a result the partition map of feature space denoted by the tree will be changed.

As mentioned earlier, each tree in our GP describes how feature space should be partitioned into hyper-cubes. A function node F_i denotes the dimension along which the input space is divided. A terminal node, on the other hand, signals that the dividing should be abandoned. Fig. 4(a) depicts a possible tree and its correspondence to the resulting partition of a two-dimensional case for the sake of better illustration. The partition scheme starts with the root node, which describes a single hyper-cube (hyper-rectangle) in the center of the input space marked by a dashed ellipsoid. This hyper-rectangle is split into two hyper-rectangles along the dimension x_1 specified by the function symbol F_1 at the root node. Subsequently, each split hyper-rectangle is performed the same so that the ultimate hyper-rectangles, which are specified by solid ellipsoids, are obtained. Theses ultimate hyper-rectangles are the representatives of the rules introduced by the tree.

To obtain the parameters of membership functions for each rule, the center and width of each hyper-cube should be calculated. Therefore two lists called *center list* and *width list* are denoted to each node. The center list and width list of a node contain the center and width of the hyper-cube which corresponds to the node. Assume, we split along dimension *i* for a parent node F_i with center list $(c_1, \ldots, c_i, \ldots, c_m)$ and width list $(\delta_1, \ldots, \delta_i, \ldots, \delta_m)$ where *m* is the dimension of the input space. The center list and width list of the left and right child node then become

$$
\begin{aligned}
c_{left} &= (c_1, \ldots, c_i - \delta_i/4, \ldots, c_m), \\
\delta_{left} &= (\delta_1, \ldots, \delta i/2, \ldots, \delta_n), \\
c_{right} &= (c_1, \ldots, c_i + \delta_i/4, \ldots, c_m), \\
\delta_{right} &= (\delta_1, \ldots, \delta i/2, \ldots, \delta_n)
\end{aligned} \tag{10}
$$

By using (10) the locations of ultimate hyper-cubes, which are interpreted as fuzzy rules, are found. The next step is to assign membership functions to the rules extracted above. We use membership functions of Gaussian form defined by (2). To extract each membership function's parameters, i.e. center and standard deviation, the following equations are used:

$$
\begin{aligned}
c &= \textit{center of hyper-cube}, \\
\sigma &= \lambda \times \textit{width of hyper-cube}.
\end{aligned} \tag{11}
$$

where λ is a design parameter.

Fig. 4(b) explains the way membership functions are assigned to a hyper-cube (hyper-rectangle) for a two-dimensional input space and $\lambda = 0.3$. As can be seen, every hyper-rectangle is assigned two membership functions, one along each input variable.

There exists a trade-off between the accuracy and complexity of a fuzzy model which should be considered in defining fitness function. The more rules the model has, the more accurate it is. On the other hand, a neuro-fuzzy system with a large number of local models is more difficult to understand and computationally less tractable than a more comprehensive one with a fewer number of rules. To take into account of this trade-off between model accuracy and the number of rules, a penalty factor, *p*, for the number of local models (number of rules), #R, is added into the fitness function. The overall fitness becomes:

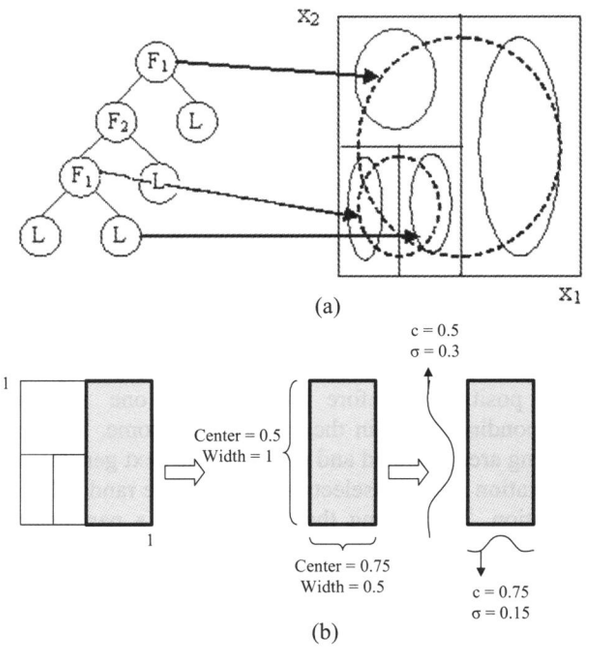


Fig. 4. (a) GP tree and corresponding partition map of input space. (b) Membership function assignment to a hyper-cube with $\lambda = 0.3$.

$$
F = \frac{1}{error + p\#R} \tag{12}
$$

error in (12) is defined as follows:

$$
error = \sqrt{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2} \Big/ \sqrt{\sum_{i=1}^{N} y_i^{\,2}} \tag{13}
$$

where *N* is number of data pairs , y_i and $\hat{y}_i$ are the i-th observed and predicted outputs respectively.

The fitness of a fuzzy model improves if adding an additional rule reduces the overall error by an amount larger than the penalty factor *p*. A small value of *p* favors input space partitions with a smaller approximation error at the cost of an increased number of rules.

4.3 Genetic Algorithm (GA)

Proposed by John Holland in the 1960s, genetic algorithms are the best known class of evolutionary algorithms. They are used so extensively that often the terms genetic algorithms and evolutionary computation are used interchangeably. In this paper a continuous GA is applied to optimize the parameters of membership functions for the structures generated by GP. Fig. 5 shows the way by which GA encodes the parameters of membership functions. Each chromosome consists of genes whose number equals to *number of rules* × *number of input variables*. Each gene encapsulates the parameters

of a Gaussian membership function, i.e. standard deviation and center.

The selection method used for GA is tournament selection like the one applied in GP. Two groups of K randomly selected chromosomes are generated and the chromosome with the best fitness function is picked up from each group. Then these two chromosomes with the offspring produced by crossover operator are sent to next generation. This process continues to fill the next generation completely.

The crossover which is used for GA is a one-pint crossover. After taking two chromosomes it randomly selects the position of a gene and then exchanges the genes positioned before the selected one with the corresponding genes in the other chromosome. Thus two offspring are generated and passed to the next generation.

Mutation operator selects a chromosome randomly in a population. Then along the chromosome a parameter is chosen again in a random manner. The value of the chosen parameter is replaced with another one based on the type of the parameter. If the parameter represents $center_i$ (the center of the i-th hyper-cube), then a random value ranged between $center_i$-$width_i/\eta$ and $center_i$+$width_i/\eta$ (η is a design parameter) will be generated and replaced. If the parameter represents $width_i$ (the width of the i-th hyper-cube), then a random value ranged between 0 and $\beta \times width_i$ will be replaced where β is a design parameter.

Since the aim of GA is to optimize the membership functions of a predetermined structure to reach the lower error, the fitness function is defined as the inversion of the model's MSE (mean square error) between the data and the model output. Thus trying to upraise the fitness value of the model, GA searches for better parameters to reduce the model error.

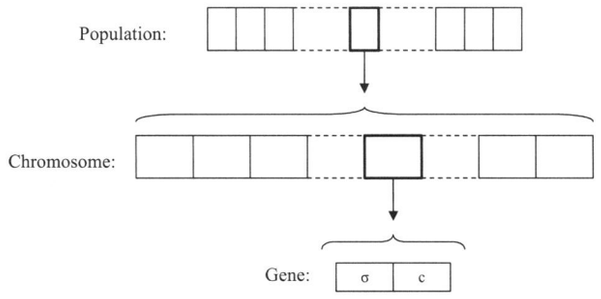


Fig. 5. Encoding of parameters by GA. Each gene includes two parameters of Gaussian membership function, i.e. standard deviation (σ) and center (c).

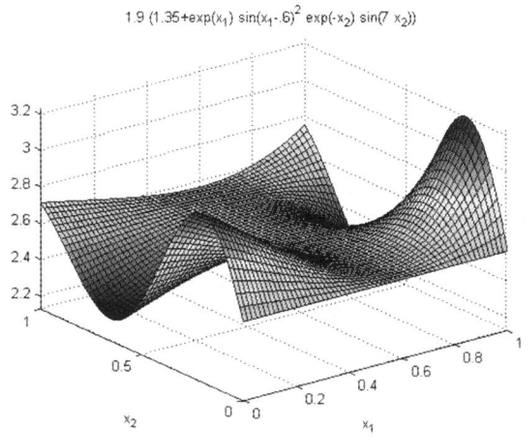


Fig 6. The nonlinear function used in example 1

5 Simulation Results

In this section we bring two non-linear systems introduced in prior studies and apply the cooperative evolution (CE) approach proposed in this paper to reach accurate models.

The following information is used for the simulations:

- General information of GA and GP
  - population size = 100 chromosomes
  - selection method = tournament selection with two competitors
  - 1 mutatuion/generation
- Specific information of GP
  - $\lambda = 0.3$
  - penalty factor (p) = 0.001
- Specific information of GA
  - $\beta = 0.5$
  - $\eta = 2$
  - number of generations = 50
  - $MSE = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \Big/ N$

where N is number of data pairs, y_i and $\hat{y}_i$ are the i-th observed and predicted outputs respectively.

5.1 Example 1

In order to show the difference between the results of simultaneous and partial optimizations in terms of the factors involved in a problem, a nonlinear system which is used by Hoffmann and Nelles in [5] is chosen. Hoffmann and Nelles proposed the following two-dimensional function as a test case for model selection of TSK fuzzy systems (Fig. 6).

$$f(x_1, x_2) = 1.9 \times (1.35 + exp(x_1) \times sin(x_1 - 0.6)^2 \times exp(-x_2) \times sin(7 \times x_2)), \quad x_1, x_2 \in [0, 1]. \tag{14}$$

Since our GP can partially optimize a fuzzy model on its own (without cooperative GA), we try to reach a fuzzy model for the preceding system in two ways; once by using only GP and another time by using GP and GA together (CE). Table I shows the average normalized root mean square error (NRMSE) of 50 runs of GP for 20 generations for the models obtained by each of two methods mentioned above, i.e. GP and CE. The results shown in Table I are attained using 400 equally spaced samples $\{(x_1, x_2, y)\}$ where $x_1 = \{ 0.05 , 0.10 , \ldots , 1 \}$ and

TABLE I
Comparison between Simultaneous and Partial Optimization for the First Numerical Example

Method	#Rule	Training NRMSE	Test NRMSE
GP	4	0.1852	0.5965
CE	3	0.0327	0.0856

TABLE II
Model Comparison for the Second Numerical Example

Method	#Rule	Training MSE	Test MSE
Wong and Chen taken from [14]	13	0.04	-
Ouyang and Lee	7	0.04	-
CE	8	0.0008	0.0007

TABLE III
Premise and Consequent Parameters for the Second Numerical Example

	MF along x1 (σ, c)	MF along x2 (σ, c)	consequent parameters $(\alpha_0^i, \alpha_1^i, \alpha_2^i)$
Rule 1	(0.1547, -0.6935)	(0.2057, 0.4840)	(1.0338, 3.8126, -1.5068)
Rule 2	(0.2232, -0.2607)	(0.1646, 0.4979)	(2.2697, 3.1532, -0.6094)
Rule 3	(0.2076, 0.0656)	(0.1986, 0.3818)	(3.4784, -0.0242, 0.2671)
Rule 4	(0.1999, 0.9446)	(0.1874, 0.4463)	(-3.1496, 0.0909, 3.2811)
Rule 5	(0.2319, -0.9223)	(0.2404, 0.5320)	(1.7295, -0.3230, 1.9938)
Rule 6	(0.1632, -0.1427)	(0.1500, 0.7500)	(-1.1090, 2.6138, -1.4708)
Rule 7	(0.2441, 0.0989)	(0.2427, 0.6359)	(-0.3988, 0.2341, -0.9821)
Rule 8	(0.2011, 0.9648)	(0.1969, 0.6184)	(3.3538, 0.5139, -3.8680)

$x_2 = \{0.05, 0.10, ..., 1\}$ that the first 300 samples are used for training and the rest are utilized for test. As we expect, the cooperation of GP and GA gives a more accurate model in comparison with what GP comes up with alone. Additionally, the model resulted by CE is simpler than what GP has suggested. In this example NRMSE is calculated as follows:

$$NRMSE = \sqrt{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \Big/ \sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2} \qquad (15)$$

where N is number of data pairs, y_i and $\hat{y}_i$ are the i-th observed and predicted outputs respectively, and $\bar{y}$ is the average of observed outputs.

5.2 Example 2

Let us consider a nonlinear function which is defined as

$$y = \sin(\pi x_1).\cos(\pi x_2) \qquad (16)$$

where $x_1 \varepsilon [-1, 1]$ and $x_2 \varepsilon [0, 1]$ (Fig. 7). The input-output data represented by $\{(x_1, x_2, y)\}$ are sampled from (16), where $x_1 = \{-1, -0.9, -0.8, ..., 1\}$ and $x_2 = \{0, 0.1, 0.2, ..., 1\}$. As a result, there are totally 231 data points for training. Additionally, 133 data points are produced randomly for testing the model. The average MSE error of 50 runs of CE for three generations is shown in Table II. The table reveals that our model outperforms the competing models with regard to the training MSE. Moreover, the convergence speed of CE, which is done in three generations, is noticeably prominent in this example.

Table III presents the parameters of membership functions and consequent parameters of each rule for the solution resulted by means of CE after three generations. Each rule has two membership functions, one of them is along x_1 and the other one is along x_2. The consequent parameters, which construct the last column of the table, have been attained with the help of least-squares method explained in (9).

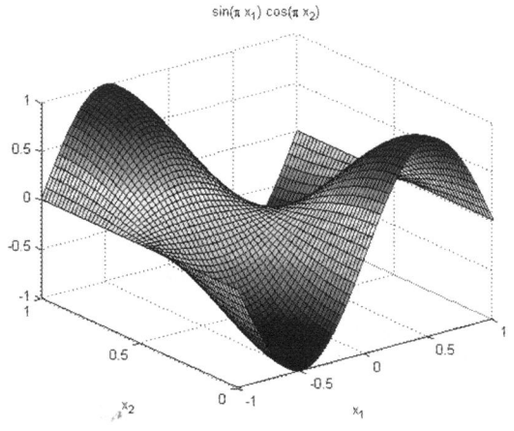


Fig 7. The nonlinear function used in example 2

6 Conclusions and Future Work

This paper presents a cooperative evolutionary approach for fuzzy modeling on ANFIS architecture. The structure and parameters of the model are generated simultaneously by imposing competition and evolution among candidate solutions. The experimental results show that the approach outperforms its rivals in finding better solutions. It should be noted that the proposed approach concerns with simplicity and accuracy of the models. Therefore, the computational burden is justified in applications for which improvements in the model accuracy or a reduction of the number of rules carry a remarkable benefit. In the future

we plan to reduce the number of membership functions and also put weights for them so that they are not considered fairly. Designing methods for obtaining the values of other parameters, such as β, η, and λ, are further optimization which can be added to this approach.

Bibliography

[1] R. Babuska and H. Verbruggen, "An overview of fuzzy modeling for control," *Control Eng. Practica, vol. 4, pp. 1593-1606, 1996.*

[2] G. Castellano and A. M. Fanelli, "An approach to structure identification of fuzzy models," *IEEE Trans. Fuzzy Syst., vol. 1, pp. 531-536, 1997.*

[3] F. Fagarasan, "A genetic-based method applied in fuzzy modeling," in *Proceedings of IEEE Int'l Conf. on Evolutionary Comp., pp. 253-257, 1996.*

[4] T. Hatanala, K. Uosaki and N. Manabe, "Structure identification in Takagi-Sugeno fuzzy modeling," in *Proceedings of IEEE Int'l Conf. on Fuzzy Syst., vol. 1, pp. 69-74, May 2002.*

[5] F. Hoffmann and O. Nelles, "Genetic programming for model selection of TSK-fuzzy systems," *Info. Sciences, vol. 136, pp. 7-28, 2001.*

[6] J.-S. Jang, "ANFIS: Adaptive-network-based fuzzy inference systems," *IEEE Trans. Syst., Man., Cybern., vol. 15, pp. 116-132, 1985.*

[7] T. Johansen, "Identification of nonlinear system structure and parameters using regime decomposition," *Automatica, vol. 31, pp. 333-350, 2000.*

[8] C. L. Karr, "Genetic algorithms for fuzzy controllers,", *AI Expert, vol. 6, pp 26-33, Feb. 1991.*

[9] C. L. Karr, L. M. Freeman and D. L Meredith, "Improved fuzzy process control of spacecraft terminal rendezvous using a genetic algorithm," in *Proceedings of Intelligent Control and Adaptive Systems Conf., vol. 1196, pp. 274-278, Feb. 1990.*

[10] J. Koza, *Genetic programming III; Darwinian invention and problem solving,* Morgan Kaufman, San. Francisco, 1999.

[11] E. H. Mamdani and S. Assilian, "Application of fuzzy algorithms for control of simple dynamic plant," in *Proceedings of Institution of Electrical Engineers, vol. 11, pp. 1585-1588, 1974.*

[12] R. Murray-Smith, A local model network approach to nonlinear modeling, Ph.D. Thesis, University of Stratholyde, 1994.

[13] O. Nelles and R. Isermann, "Basis Fuction networks for interpolation of local linear models, " in *Proceedings of IEEE Conf. on Decision and control, pp. 470-475, 1996.*

[14] C.-S. Ouyang and S.-J. Lee, "A hybrid algorithm for structure identification of neuro-fuzzy modeling," in *Proceedings of IEEE Int'l Conf. on Syst., Man., Cybern., vol. 5, pp. 3611-3616, Oct. 2000.*

[15] S. E. Papadakis and J. B. Theocharis, "A GA-based fuzzy modeling approaches for generating TSK models," *Fuzzy Sets Syst., vol. 131, pp. 121-152, 2002.*

[16] W. Pedrycz and M. Reformat, "Evolutionary fuzzy modeling," *IEEE Trans. Fuzzy Syst., vol. 11, pp. 652-665, Oct. 2003.*

[17] C. A. Pena-Reyes and M. Sipper, "Fuzzy CoCo: A coevolutionary coevolutionary approach to fuzzy modeling," *IEEE Trans. Fuzzy Syst., vol. 9, no. 5, pp. 727-737, Oct. 2001.*

[18] M. Russo, "Genetic fuzzy learning," *IEEE Trans. Evolutionary Comput., vol. 4, pp. 259-273, June 2000.*

[19] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets Syst., vol. 28, pp. 15-33, 1988.*

[20] C.-T. Sun, "Rule-base structure identification in an adaptive-network-based fuzzy inference system," *IEEE Trans. Fuzzy Syst., vol. 2, pp. 64- 73*, Feb. 1994.

[21] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man., Cybern., vol. 23, pp. 665-685, 1993.*

[22] M. Tomassini, "A survey of genetic algorithms," *World Scientific, vol. 3, pp. 87-118, 1995.*

[23] S. G. Tzafestas and K. C. Zikidis, "NeuroFAST: on-line neuro-fuzzy ART-based structure and parameter learning TSK model," *IEEE Trans. Syst. Man., Cybern., vol. 31, pp. 797-802, Oct. 2001.*

[24] C.-C. Wong and C.-C. Chen, "A hybrid clustering and gradient descent approach for fuzzy modeling," *IEEE Trans. Syst., Man, Cybern. B, vol. 29, pp. 686-693, Dec. 1999.*

[25] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and processes," *IEEE Trans. Syst. Man. Cybern. SMC-3, pp. 28-44, Jan. 1973.*