Available online at www.sciencedirect.com
SCIENCE DIRECT®
Fuzzy Sets and Systems 152 (2005) 617–635
FUZZY sets and systems
www.elsevier.com/locate/fss

Genetic fuzzy logic controller: an iterative evolution algorithm with new encoding method

Yu-Chiun Chiou[a,*], Lawrence W. Lan[b]

[a] *Department of Traffic and Transportation Engineering and Management, Feng Chia University, Taichung 40724, Taiwan, ROC*
[b] *Institute of Traffic and Transportation, National Chiao Tung University, Taipei 10012, Taiwan, ROC*

Received 23 July 2003; received in revised form 26 August 2004; accepted 23 November 2004
Available online 18 December 2004

Abstract

Logic rules and membership functions are two key components of a fuzzy logic controller (FLC). If only one component is learned, the other one is often set subjectively thus can reduce the applicability of FLC. If both components are learned simultaneously, a very long chromosome is often needed thus may deteriorate the learning performance. To avoid these shortcomings, this paper employs genetic algorithms to learn both logic rules and membership functions sequentially. We propose a bi-level iterative evolution algorithm in selecting the logic rules and tuning the membership functions for a genetic fuzzy logic controller (GFLC). The upper level is to solve the composition of logic rules using the membership functions tuned by the lower level. The lower level is to determine the shape of membership functions using the logic rules learned from the upper level. We also propose a new encoding method for tuning the membership functions to overcome the problem of too many constraints. Our proposed GFLC model is compared with other similar GFLC, artificial neural network and fuzzy neural network models, which are trained and validated by the same examples with theoretical and field-observed car-following behaviors. The results reveal that our proposed GFLC has outperformed.

Keywords: Genetic algorithms; Genetic fuzzy logic controller; Artificial neural network; Fuzzy neural network; Car-following behaviors

* Corresponding author. Tel.: +886 4 2451 7250X 4666; fax: +886 4 2452 0678.
E-mail address: ycchiou@fcu.edu.tw (Y.-C. Chiou).

0165-0114/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2004.11.011

1. Introduction

Fuzzy logic controller (FLC), an expert system based on IF-THEN fuzzy rules, has the advantages of treating ambiguous or vague aspects of human perception and judgment, with which a non-fuzzy expert system normally cannot deal. Since fuzzy control system has excellent performance in data mapping, FLC has been widely applied in various areas. In traffic and transportation studies, for instance, FLC has been used for selecting the transportation investment projects and for modeling trip generation, trip distribution, modal split, and route choice in transportation planning. It is also being used in traffic controls and related studies including aircraft control, ship loading/unloading control, intersection signal control, accident analysis/prevention, and level of service evaluation [28].

Learning the logic rules and tuning the membership functions are the two key components for a FLC system. Genetic algorithms (GAs) have been proven suitable for solving both combinatory optimization and parameter optimization problems (i.e., rules selection and parameters calibration). Employing GAs to construct a FLC system with learning process from examples, hereafter abbreviated as genetic fuzzy logic controller (GFLC), not only can avoid the bias caused by subjective settings of logic rules and membership functions but also can greatly enhance the control performance. Hence, GFLC has received intensive attentions from researchers. In traffic and transportation applications, for instance, Wang and Mendel [30] employ GFLC to estimate the number of air passengers traveling between major industrial cities and given regions. Bonissone et al. [1] apply GFLC to control the velocity of a freight train. Hwang [14] applies GFLC to optimize trip time and energy consumption of a high-speed railway. All of these works found that GFLC has performed quite effectively. Most previous studies, however, have employed GAs either to calibrate the membership functions with preset logic rules or to select the logic rules with given membership functions. More recent studies have been devoted to incorporating both either simultaneously or sequentially. However, if the parameters of membership functions are encoded directly for tuning, there would be too many constraints thus may require extremely large searching space and deteriorate the learning performance.

This paper proposes a bi-level iterative evolution algorithm to learn logic rules and membership functions in sequence without subjectively presetting both of them. To overcome the shortcoming of direct parameters encoding, this paper also proposes a new encoding method that can tune the membership functions more effectively. In order to demonstrate the practicability of our proposed GFLC model and to compare the control performance with other models, including two similar GFLC models, artificial neural network model, and fuzzy neural network model, both theoretical and field-observed car-following behaviors are examined and validated.

2. The fuzzy logic controller

2.1. The components of fuzzy logic controller

The FLC system, first proposed by Zadeh [33], uses fuzzy logic rules to form a control mechanism to approximate expert perception and judgment under given conditions. The system is also known as fuzzy control system or fuzzy inference system or approximate reasoning or expert system. Its framework can

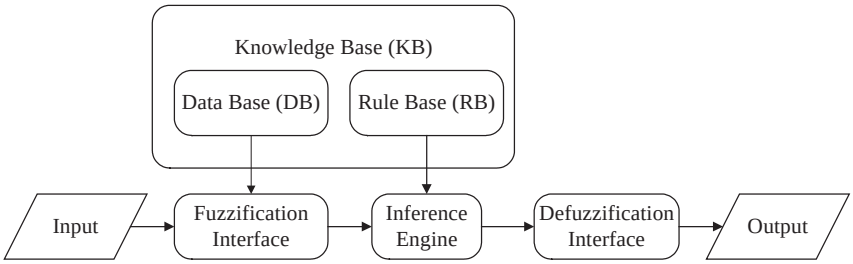


Fig. 1. Framework of fuzzy logic controller.

be depicted in Fig. 1, in which four basic components are briefly explained:

(1) *Rule base* (*RB*)
The RB is composed of finite IF–THEN rules, from which an inference mechanism is formed. A standard form of RB with m logic rules is represented as:

Rule 1: IF $x_1 = A_{11}$ and $x_2 = A_{12}$ and ... and $x_n = A_{1n}$ THEN $y = B_1$
Rule 2: IF $x_1 = A_{21}$ and $x_2 = A_{22}$ and ... and $x_n = A_{2n}$ THEN $y = B_2$
⋮
Rule m: IF $x_1 = A_{m1}$ and $x_2 = A_{m2}$ and ... and $x_n = A_{mn}$ THEN $y = B_m$,

where $x_1, \ldots, x_n$ are state variables and y is control variable. $A_{i1}, \ldots, A_{in}$ and B_i $(i = 1, \ldots, m)$ are, respectively, the linguistic variables for $x_1, \ldots, x_n$ and y in the universal of discourse of $U_1, \ldots, U_n$ and V. Taking the driving speed as an example, the linguistic degrees can be very fast, fast, normal, slow and very slow.

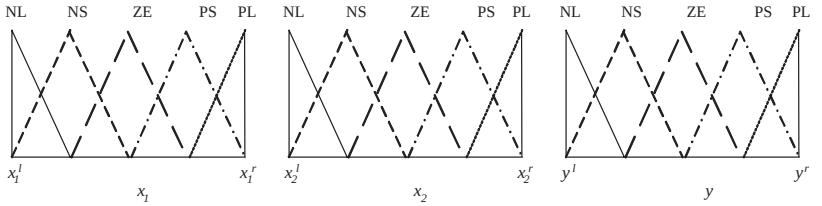
(2) *Data base* (*DB*)
The DB is formed by the specific membership functions of linguistic variables $A_{i1}, \ldots, A_{in}$ and B_i that transform crisp inputs into fuzzy ones. Triangle, trapezoid and bell-shaped membership functions are commonly used.

(3) *Inference engine*
The operators within the logic rules form the inference engine. Generally, logic rules use AND (taking minimum value) or OR (taking maximum value) as connecting operators between state variables.

(4) *Defuzzification*
For making a decision, defuzzification is the synthesis of inference results of all activated logic rules into crisp outputs. Maximum membership method, center of average method and center of gravity method are commonly used.

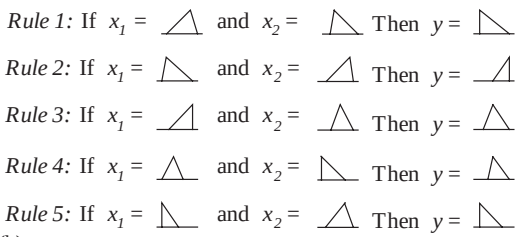
2.2. Genetic fuzzy logic controller

The methods used in the last two components of a FLC system, the inference engine and defuzzification, are rather consistent in most applications. In contrast, the methods for formulating the RB and DB of a FLC system are more subjective and complex. Due to the excellent performance of GAs in solving both combinatory optimization and parameter optimization problems, there have been several hundreds



Rule 1: If $x_1 = NL$ and $x_2 = PS$ Then $y = NS$
Rule 2: If $x_1 = NS$ and $x_2 = NL$ Then $y = PS$
Rule 3: If $x_1 = NS$ and $x_2 = PS$ Then $y = NL$
Rule 4: If $x_1 = PS$ and $x_2 = NL$ Then $y = NL$
Rule 5: If $x_1 = PL$ and $x_2 = NS$ Then $y = NS$
Rule 6: If $x_1 = PS$ and $x_2 = PL$ Then $y = PL$
Rule 7: If $x_1 = ZE$ and $x_2 = PL$ Then $y = PS$
Rule 8: If $x_1 = PS$ and $x_2 = ZE$ Then $y = ZE$

(a)



(b)

Fig. 2. Types of knowledge base [6]: (a) Descriptive knowledge base; (b) Approximate fuzzy rule base.

of studies applying GAs to the construction of FLC. A thorough and comprehensive review of GFLC literatures can be found in [4,6]. GFLC related studies can generally be divided into four categories: (1) use of GAs to tune the membership functions under a given set of logic rules; (2) use of GAs to select the logic rules with known membership functions; (3) use of GAs to learn both logic rules and membership functions simultaneously; (4) use of GAs to learn both logic rules and membership functions sequentially. Delineate as follows:

(1) *Tuning membership functions only*

Depending on the types of knowledge base, descriptive knowledge base and approximate fuzzy rule base can further be identified as shown in Fig. 2 [6]. The former assumes that membership functions of a specific state/control variable are the same in all logic rules; the latter allows that the membership functions of the same state/control variable can be varied in different logic rules. Therefore, tuning the membership functions of descriptive knowledge base implies calibrating the parameters of membership functions with a fixed number of linguistic degrees for $A_{i1}, \ldots, A_{in}$ and B_i [11,16–18,25]. Tuning the membership functions of approximate fuzzy rule base means calibrating the parameters in membership functions $A_{i1}, \ldots, A_{in}$ and B_i for each logic rule [8,12]. The major disadvantages of descriptive knowledge base are the rapid increase in chromosome length as linguistic degrees increase and it needs to consider the reasonableness of relative values among parameters. Nonetheless, this structure has the advantages that the length of chromosome is not related to the number of logic rules and that the meaning of calibrated logic rules could be implicitly interpreted as an expert's judgment and decision. On the contrary, approximate fuzzy rule base has the disadvantages that the length of chromosome grows rapidly as the number of logic rules increases and that it is sometimes difficult to interpret the calibrated logic rules as an expert's judgment and

decision. But there is no need to consider the reasonableness of relative values among parameters. The most common shapes of membership functions are triangular [5,16–18,25], trapezoidal [11,18] or Gaussian functions [10]. Each shape function has some parameters to be coded and tuned. For instance, the coordinates of cortex and two anchors of a triangular shape need to be determined. To work with reasonable distributions between linguistic fuzzy sets, tuning ranges or more specific shapes, such as fixed width of triangle base, fixed overlapping width between fuzzy sets, and isosceles triangle, are normally assumed. As an example, Gurocak [10] specifies tuning ranges for shifting the peak locations of the fuzzy sets according to a series of binary genes.

(2) *Selecting logic rules only*

Three commonly used encoding methods for selecting the logic rules are identified in the literature. The first method, in Lekova et al. [21], uses one gene, with a binary value, to represent inclusion or non-inclusion of a specific logic rule. The chromosome length depends on the number of potential logic rules. The second method, in Chin and Qi [2], uses one gene to represent the first part (premise) of a specific logic rule and the following genes to represent the latter part (linguistic degrees of control variable) of the same logic rule. The third method, in Thrift [29], uses one gene to represent each logic rule and the value of each gene indicates the linguistic degree of control variable for the corresponding logic rule. Contrasting to the first method, the second and third methods impose a constraint that a specific premise cannot map to different linguistic degrees of control variables. The strength of the first method is that all possible permutations of logic rules can be considered; its weakness is that the chromosome might be too long. The strength and weakness of the second and third methods are just contrary to the first method.

(3) *Learning logic rules and membership functions simultaneously*

There are two categories of methods to learn logic rules and membership functions simultaneously. The methods of the first category employ genetic learning algorithm to learn both logic rules and membership functions. For example, in [13,26,29,32], they use part of a chromosome to represent the composition of logic rules and use the remaining part to represent the shapes of membership functions. For instance, Homaifar and McCormick [13] adopt Thrift [29] encoding method by using part of a chromosome to represent the composition of logic rules and using each gene of the remaining part to represent the triangle base of each membership function. Xiong and Litz [32] use binary coding of rule premises and use integer coding of the two anchors of triangular membership functions with the constraint that the right anchor of a membership function is coincided with the left anchor of the next adjacent membership function. Herrera et al. [12] use a chromosome to represent the membership functions of all variables in a logic rule. It is the structure of approximate fuzzy rule base, in which the fitness of GAs cannot be evaluated as the control performance; other criteria must be designed to represent the fitness of chromosome. The methods of the second category use the hybrid learning algorithms by combining GAs with other algorithms to learn both logic rules and membership functions. For instance, Wang and Yen [31] use GAs to solve the combination of logic rules and employs Kalman filter to tune the shapes of membership functions. Lin [22] constructs a genetic algorithm-based neural fuzzy system (GA-NFS) by employing GAs to tune the membership functions in the precondition part of fuzzy rules and using least-squares estimation to tune the parameters in the consequent part.

(4) *Learning logic rules and membership functions sequentially*

Both two- and three-stage procedures are found in learning the logic rules and membership functions sequentially. A two-stage procedure, proposed by Karr [16], uses GAs to learn logic rules and then

uses another GAs to tune the membership functions. Kinzel et al. [19] and Cordon et al. [5] establish a three-stage procedure by presetting an initial pool of logic rules, then using GAs to select logic rules from the pool, and finally using another GAs to tune the membership functions. Chung et al. [3] propose another three-stage hybrid learning algorithm. In the first stage, the fuzzy ART algorithm is used to do fuzzy clustering in the input/output spaces according to supervised training data. In the second stage, GA is used to select logic rules by associating input clusters and output clusters. In the third stage, the backpropagation algorithm is used to tune the membership functions.

In sum, logic rules and membership functions are two essential components of a FLC system. If only one component is learned, the other one must be set subjectively; as a consequence, the applicability of FLC could be greatly reduced. If both components are learned simultaneously, a very long chromosome is often needed; thus it could deteriorate the learning performance. To avoid these shortcomings, this paper employs GAs to learn both logic rules and membership functions sequentially. Because the learning results of approximate fuzzy rule base can hardly be interpreted as actual experts' decision behaviors, this paper adopts the structure of descriptive knowledge base.

3. Model formulation

3.1. Encoding method for logic rules

The encoding method proposed by Thrift [29] can effectively curtail the length of chromosome and thus save the computer memory and searching space, this paper also employs this encoding method for selecting the logic rules. Taking two state variables and one control variable as an example, if each variable has five linguistic degrees (NL: negative large, NS: negative small, ZE: zero, PS: positive small, PL: positive large), then the chromosome length is 25. Genes take the integers from 0 to 5, where 0 represents the exclusion of the rules; other numbers indicate the inclusion of the rules and the five linguistic degrees of control variable. This encoding method is depicted in Fig. 3. A chromosome with gene sequence of 0002040010000001000030000, for example, will represent five logic rules being selected:

Rule 1: IF $x_1 = NL$ and $x_2 = PS$ THEN $y = NS$.
Rule 2: IF $x_1 = NS$ and $x_2 = NL$ THEN $y = PS$.
Rule 3: IF $x_1 = NS$ and $x_2 = PS$ THEN $y = NL$.
Rule 4: IF $x_1 = PS$ and $x_2 = NL$ THEN $y = NL$.
Rule 5: IF $x_1 = PL$ and $x_2 = NL$ THEN $y = ZE$.

3.2. Encoding method for membership function

Consider a triangle fuzzy number and let parameters c_k^r, c_k^c and c_k^l, respectively, represent the coordinates of right anchor, cortex and left anchor of kth linguistic degree. Then 15 parameters need to be calibrated for a variable with five linguistic degrees. Additional constraints are imposed: $c_k^r \geqq c_k^c \geqq c_k^l$ for any fuzzy number with the same degree and $c_k^r \geqq c_{k-1}^r$, $c_k^c \geqq c_{k-1}^c$, $c_k^l \geqq c_{k-1}^l$ for fuzzy numbers with different degrees. Thus, if we employ GAs to tune the aforementioned parameters directly, the search performance will deteriorate significantly as a result of incorporating all the constraints. To overcome this problem, we propose a new encoding method for tuning the membership functions.

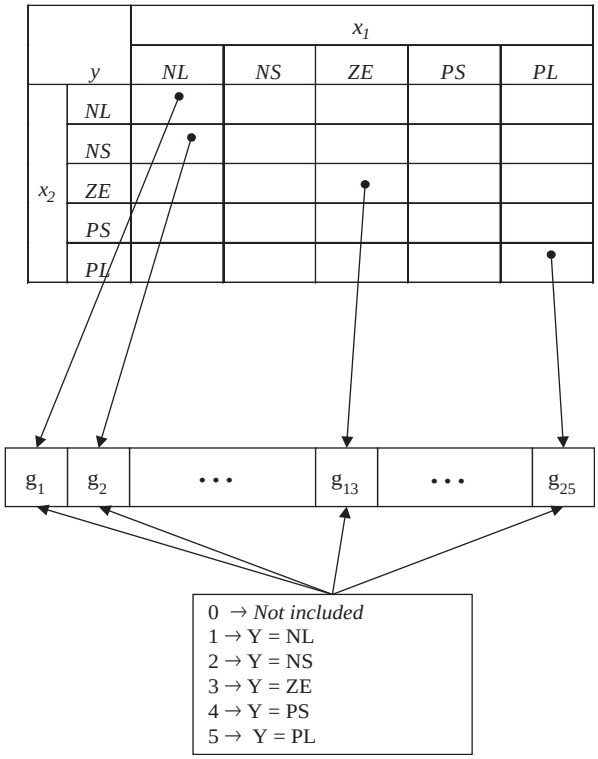


Fig. 3. Encoding method for logic rules.

Without loss of generality, we assume that the first and last degrees of fuzzy numbers are left- and right-skewed triangles, respectively and that the others are isosceles triangles as shown in Fig. 4. Therefore, a variable with five linguistic degrees has eight parameters to be calibrated and their orders are:

$$c_{\max} = c_5^c = c_5^r \geqslant c_4^r \geqslant \begin{matrix} c_5^l \\ c_3^r \end{matrix} \geqslant \begin{matrix} c_4^l \\ c_2^r \end{matrix} \geqslant \begin{matrix} c_3^l \\ c_1^r \end{matrix} \geqslant c_2^l \geqslant c_1^c = c_1^l = c_{\min}, \tag{1}$$

$$c_k^c = \frac{(c_k^r + c_k^l)}{2}, \quad k = 2, 3, 4, \tag{2}$$

where $c_{\max}$ and $c_{\min}$ are the maximum and minimum values of the variable, respectively. The orders between c_5^l and c_3^r, c_4^l and c_2^r, c_3^l and c_1^r are indeterminate. In order to tune these eight parameters, nine position variables $r_1 \sim r_9$ are designed as follows:

$$c_2^l = c_{\min} + r_1 \times \theta, \tag{3}$$

$$c_1^r = c_2^l + r_2 \times \theta, \tag{4}$$

$$c_3^l = c_2^l + r_3 \times \theta, \tag{5}$$

$$c_2^r = \max\{c_1^r, c_3^l\} + r_4 \times \theta, \tag{6}$$

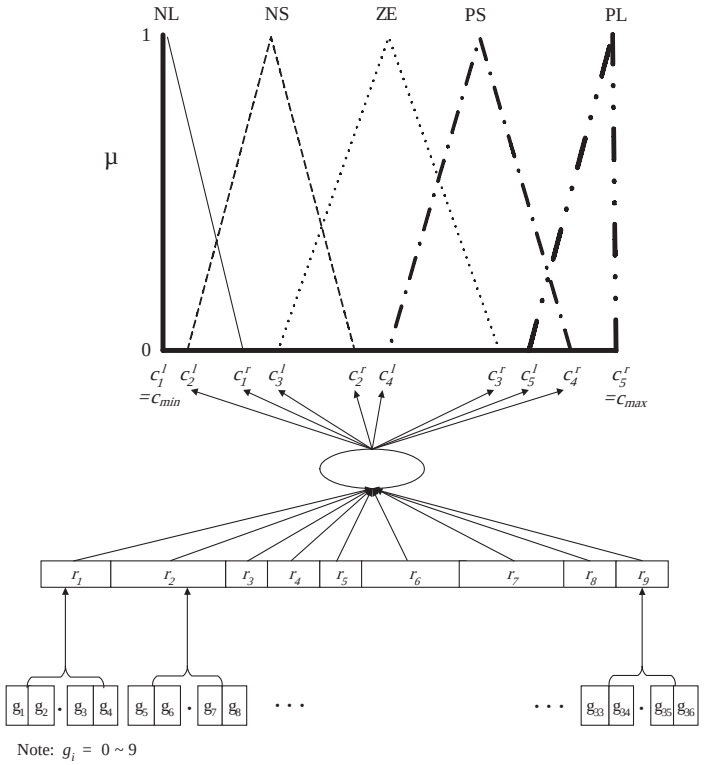


Fig. 4. Encoding method for membership functions.

$$c_4^l = \max\{c_1^r, c_3^l\} + r_5 \times \theta, \tag{7}$$

$$c_3^r = \max\{c_2^r, c_4^l\} + r_6 \times \theta, \tag{8}$$

$$c_5^l = \max\{c_2^r, c_4^l\} + r_7 \times \theta, \tag{9}$$

$$c_4^r = \max\{c_3^r, c_5^l\} + r_8 \times \theta, \tag{10}$$

where

$$\theta = \frac{(c_{\max} - c_{\min})}{\sum_{i=1}^{9} r_i}.$$

To achieve two significant digits, each position variable is represented by four real-coding genes also depicted in Fig. 4. The maximum value of the position variables is 99.99 and the minimum value is 0. Thus, in the example of two state variables and one control variable (each with five linguistic degrees), the chromosome is composed of 108 genes.

3.3. Genetic operators

Because the genes in our GFLC model are not encoded binary, simple genetic algorithms [9] cannot be used. Instead, we employ the max–min-arithmetical crossover proposed by Herrera et al. [11] and the non-uniform mutation proposed by Michalewicz [24]. A brief description is given below.

(1) Max–min-arithmetical crossover
Let $G_w^t = \{g_{w1}^t, \ldots, g_{wk}^t, \ldots, g_{wK}^t\}$ and $G_v^t = \{g_{v1}^t, \ldots, g_{vk}^t, \ldots, g_{vK}^t\}$ be two chromosomes selected for crossover, the following four offsprings will be generated [11]:

$$G_1^{t+1} = aG_w^t + (1-a)G_v^t, \tag{11}$$

$$G_2^{t+1} = aG_v^t + (1-a)G_w^t, \tag{12}$$

$$G_3^{t+1} \text{ with } g_{3k}^{t+1} = \min\{g_{wk}^t, g_{vk}^t\}, \tag{13}$$

$$G_4^{t+1} \text{ with } g_{4k}^{t+1} = \max\{g_{wk}^t, g_{vk}^t\}, \tag{14}$$

where a is a parameter ($0 < a < 1$) and t is the number of generations.

(2) Non-uniform mutation
Let $G_t = \{g_1^t, \ldots, g_k^t, \ldots, g_K^t\}$ be a chromosome and the gene g_k^t be selected for mutation (the domain of g_k^t is $[g_k^l, g_k^u]$), the value of g_k^{t+1} after mutation can be computed as follows:

$$g_k^{t+1} = \begin{cases} g_k^t + \Delta(t, g_k^u - g_k^t) & \text{if } b = 0, \\ g_k^t - \Delta(t, g_k^t - g_k^l) & \text{if } b = 1, \end{cases} \tag{15}$$

where b randomly takes a binary value of 0 or 1. The function $\Delta(t, z)$ returns a value in the range of $[0, z]$ such that the probability of $\Delta(t, z)$ approaches to 0 as t increases:

$$\Delta(t, z) = z(1 - r^{(1-t/T)^h}), \tag{16}$$

where r is a random number in the interval $[0, 1]$, T is the maximum number of generations and h is a given constant. In Eq. (15), the value returned by $\Delta(t, z)$ will gradually decrease as the evolution progresses.

3.4. The iterative evolution algorithm

Similar to a bi-level mathematical programming, we propose a bi-level iterative evolution algorithm in selecting the logic rules and tuning the membership functions for a genetic fuzzy logic controller (GFLC). The upper level is to solve the composition of logic rules using the membership functions tuned by the lower level. The lower level is to determine the shape of membership functions using the logic rules learned from the upper level.

Consider a FLC with n state variables $x_1, x_2, \ldots, x_n$ and one control variable y, each with $d_1, d_2, \ldots, d_n$ and d_{n+1} linguistic degrees. Assume that the membership functions of all linguistic degrees to be triangle-shaped. Our proposed iterative evolution algorithm is structured as follows:

Step 0: Initialization: $s = 1$.

Step 1: Selecting logic rules.

Step 1.1: Generating an initial population with p chromosomes. Each chromosome has $\prod_{i=1}^{n} d_i$ genes, and each gene randomly takes one integer from $[0, d_{n+1}]$.
Step 1.2: Calculating the fitness values of all chromosomes based on incumbent membership functions.
Step 1.3: Selection.
Step 1.4: Crossover.
Step 1.5: Mutation.
Step 1.6: Testing the stop condition. The stop condition is set based on whether the mature rate (the proportion of same chromosome in a population) has reached a given constant η. If so, proceed to Step 2; otherwise go to Step 1.2.

Step 2: Tuning membership functions.
Step 2.1: Generating an initial population with p chromosomes. Each chromosome has $36(n+1)$ genes and each gene randomly takes one integer from $[0, 9]$.
Step 2.2: Calculating the fitness values of all chromosomes based on the incumbent logic rules.
Step 2.3: Selection.
Step 2.4: Crossover.
Step 2.5: Mutation.
Step 2.6: Testing the stop condition. Let f_s be the largest fitness among the population for the sth evolution epoch. The stop condition is set based on whether the mature rate has reached a given constant η. If so, proceed to Step 3 and let $s = s + 1$; otherwise go to Step 2.2.

Step 3: Testing the stop condition. If $(f_{s+1} - f_s) \leqq \varepsilon$, where ε is an arbitrary small number, then stop. Incumbent logic rules and membership functions are the optimal learning results. Otherwise, go to Step 1.

4. Validations

In order to validate and to compare the performance of our GFLC model with other models including two similar GFLC models (proposed by Homaifar and McCormick [13], hereafter GFLC1 model and by Gurocak [10], hereafter GFLC2 model), artificial neural network (ANN) and fuzzy neural network (FNN), both theoretical and field-observed car-following behaviors are examined.

4.1. The theoretical car-following behaviors

Jiang et al. [15] proposed a theoretical car-following model as follows:

$$\ddot{x}_{n+1}(t) = k\{V[\dot{x}_n(t) - \dot{x}_{n+1}(t)] - \dot{x}_{n+1}(t)\} + \frac{\alpha[\dot{x}_{n+1}(t)]^m}{[x_n(t) - x_{n+1}(t)]^l}[\dot{x}_n(t) - \dot{x}_{n+1}(t)], \tag{17}$$

where $\ddot{x}_{n+1}(t)$: acceleration/deceleration of following vehicle at time t. $\dot{x}_n(t)$ and $\dot{x}_{n+1}(t)$: speeds of leading and following vehicles at time t, respectively. $x_n(t)$ and $x_{n+1}(t)$: positions of leading and following vehicles at time t, respectively, k, α, m, l: parameters. $V[\dot{x}_n(t) - \dot{x}_{n+1}(t)]$: legal speed of the following vehicle, defined as $V[\dot{x}_n(t) - \dot{x}_{n+1}(t)] = V_f\{1 - \exp[1 - \exp(|C_j|/V_f(\dot{x}_n(t) - \dot{x}_{n+1}(t)/H_j - 1))]\}$ in which V_f denotes the free-flow speed, C_j is the wave speed at jam density and H_j represents the jam spacing [7]. According to Del Castillo and Benitez [7] and Jiang et al. [15], the parameters of the theoretical car-following model are set as $k = 0.035$, $V_f = 86.4$ km/h, $|C_j| = 11.92$ km/h, $H_j = 6.18$ m, $\alpha = 100$,

$m = 1, l = 2$. To generate sufficient learning and validating samples, we use a random number generator to produce a series of accelerations/decelerations for the leading vehicle in every second according to a uniform distribution $[-3, 3]$. The initial conditions for the leading and following vehicles are set as $x_n(0) = 50\,\text{m}$, $x_{n+1}(0) = 5\,\text{m}$ and $\dot{x}_n(0) = \dot{x}_{n+1}(0) = 20\,\text{m/s}$. A total of 100-s car-following samples are generated for learning and another 50-s data for validating.

4.2. Experimental results from GFLC car-following model

The GFLC model aims to minimize the difference between predicted and theoretical behaviors of the car-following model in Eq. (17). The difference is measured by the energy function (E), a total sum of squared errors, defined as

$$E = \frac{1}{2}\sum_{i=1}^{N}[\hat{y}_i - y_i]^2, \tag{18}$$

where $\hat{y}_i$ and y_i, respectively, represent the predicted and theoretical accelerations/decelerations of the following vehicle for the ith training sample. N is the total number of training samples. The fitness function of GAs is then set as $f = 1/E$.

The car-following model in Eq. (17) has three state variables: relative distance ($\text{RD} = x_n(t) - x_{n+1}(t)$), relative speed ($\text{RS} = \dot{x}_n(t) - \dot{x}_{n+1}(t)$), and following vehicle speed ($\text{FS} = \dot{x}_{n+1}(t)$) and one control variable: following vehicle acceleration ($\text{FA} = \ddot{x}_{n+1}(t)$). Assume that all variables have five linguistic degrees represented by triangle membership functions. This makes a total of 625 potential logic rules. With one gene for each rule, there are 125 genes in a chromosome. Thus a total of 36 position parameters need to be calibrated for tuning the membership functions. With four genes for each parameter, a total of 144 genes are in a chromosome. The parameters of the GFLC car-following model are set as population size = 100, crossover rate = 0.9, $a = 0.3$, $h = 0.5$, $\eta = 80\%$, $\varepsilon = 0.001$. The center of gravity method is employed for defuzzification.

The training and validating results of GFLC car-following model for various mutation rates are reported in Table 1. It indicates that GFLC best performs at the mutation rate of 0.30 with corresponding values of energy function (E) for training and validating 2.55 and 1.07, respectively. Fig. 5 further depicts the learning process of GFLC at the mutation rate of 0.3. Note that GFLC converges after four iterative evolutions with 679 generations progressed. The value of energy function decreases from 113.07 to 2.55.

4.3. Comparison with GFLC1, GFLC2, ANN and FNN models

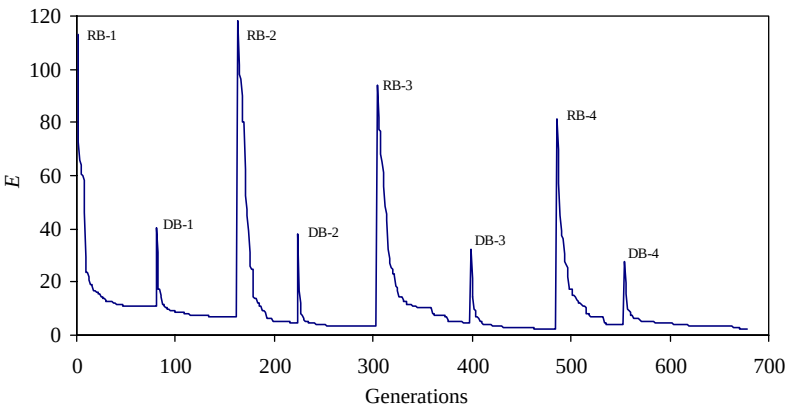
The same theoretical car-following data generated from Eq. (17) is also used to train and validate the GFLC1, GFLC2, ANN and FNN car-following models. A brief introduction of these models and the comparison results are given below.

GFLC1 simultaneously learns the logic rules and membership functions. For the logic rules encoding, GFLC1 uses part of a chromosome to represent the composition of logic rules by adopting Thrift's [29] encoding method. While for the membership functions encoding, GFLC1 uses each gene of the remaining part of a chromosome to represent the triangle base of each membership function. The encoding method of GFLC1 is depicted in Fig. 6. GFLC2 only tunes the Gaussian membership functions with preset logic rules by specifying the tuning ranges for shifting the peak locations of the fuzzy sets according to a

Table 1
The results of GFLC, GFLC1 and GFLC2 with various mutation rates (*Pm*)

Models	Performance	*Pm*								
		0.01	0.03	0.05	0.07	0.10	0.20	0.30	0.40	0.50
GFLC	No. of generations	716	606	643	653	722	655	679	600[a]	600[a]
	E (Training)	7.17	8.23	7.28	7.15	7.07	2.91	2.55[b]	5.81	9.67
	E (Validating)	4.84	5.67	4.64	4.05	4.23	1.55	1.07[b]	3.21	4.78
GFLC1	No. of generations	112[a]	141	187	135	185	157	192	173	168
	E (Training)	33.02	29.03	33.22	11.73	8.30	6.00[b]	15.59	16.68	12.15
	E (Validating)	30.97	24.38	13.32	5.23	4.76	3.78[b]	15.61	15.10	9.02
GFLC2	No. of generations	719	836	671	876	706	635	782	654[a]	734
	E (Training)	24.25	14.05	12.95	12.42	9.87	6.11	7.25	5.59	4.16[b]
	E (Validating)	19.21	10.78	8.01	8.21	5.34	3.65	5.13	3.75	2.60[b]

[a]Indicates the most efficient performance of *Pm* for each model.
[b]Indicates the most effective performance of *Pm* for each model.

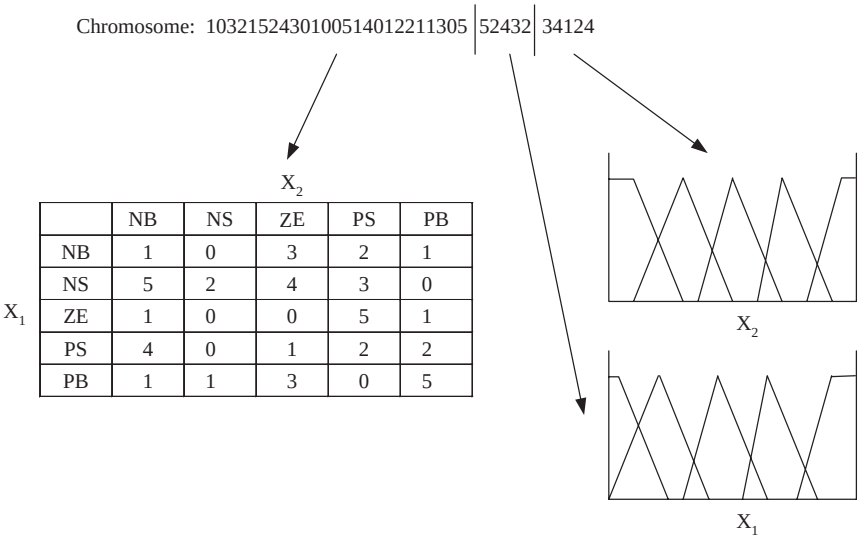


Note: RB-# represents the #th evolution of selecting logic rules. DB-# represents the #th evolution of tuning membership functions.

Fig. 5. The learning process of GFLC car-following model.

series of binary genes. To facilitate the comparison, the subjectively preset logic rules of GFLC2 are determined by GAs using Thrift's encoding method [29]. Moreover, the Gaussian membership functions are substituted by triangular ones. The tuning ranges and the encoding method of GFLC2 are shown in Fig. 7.

The results of GFLC1 and GFLC2 are also reported in Table 1, which show that GFLC1 best performs at the mutation rate of 0.20, with training and validating energy function (*E*) of 6.00 and 3.78, respectively. Obviously, our proposed GFLC model performs better than the GFLC1; however, GFLC1 converges with only 157 generations in the training process. It indicates that GFLC1 is more efficient than our GFLC



X1 \ X2	NB	NS	ZE	PS	PB
NB	1	0	3	2	1
NS	5	2	4	3	0
ZE	1	0	0	5	1
PS	4	0	1	2	2
PB	1	1	3	0	5

Fig. 6. The encoding method for logic rules and membership functions of GFLC1 [13].

model. Likewise, GFLC2 best performs at the mutation rate of 0.50, with training and validating energy function of 4.16 and 2.60, respectively, which are greater than those of our GFLC model. Besides, GFLC2 converges with 734 generations in the training process. It suggests that our GFLC model is superior to GFLC2 both in effectiveness and efficiency.

In constructing the ANN car-following model, the package *NeuralWorks Predict* is employed. This software package can automatically choose the fittest network structure. The results show that the values of energy function for training and validating are 3.62 and 1.48, respectively, which are slightly higher than those of our GFLC model. In constructing the FNN car-following model, a network structure proposed by Lan et al. [10] with five layers (3, 15, 5, 5, 1) is used. The values of energy function for training and validating are 3.04 and 1.83, respectively, which are also slightly higher than those of GFLC.

The predicted accelerations/decelerations of the following vehicle by GFLC, GFLC1, GFLC2, ANN and FNN models are compared with theoretical data (Fig. 8). Note that these five models can predict theoretical car-following behaviors rather precisely and that GFLC has outperformed. The shapes of membership functions tuned by GFLC, GFLC1, GFLC2 and FNN are depicted in Fig. 9. Note that the shapes of membership functions learned from GFLC, GFLC1 and GFLC2 and FNN models are not exactly the same. All of the three GFLC models have learned both the combination of logic rules and the shapes of membership functions. FNN, working as a partial black box, has only identified the shapes of membership functions without knowing exactly the combination of logic rules. ANN, working as a total black box, has not obtained any information about the logic rules and membership functions.

4.4. Discussions

We find that a total of 114 logic rules have been selected by GFLC, 124 logic rules by GFLC1 and 107 logic rules by GFLC2, which are too large to be interpreted as a whole. There could be three reasons for choosing so many logic rules. First, the control performance of GFLC models is synthesized by many

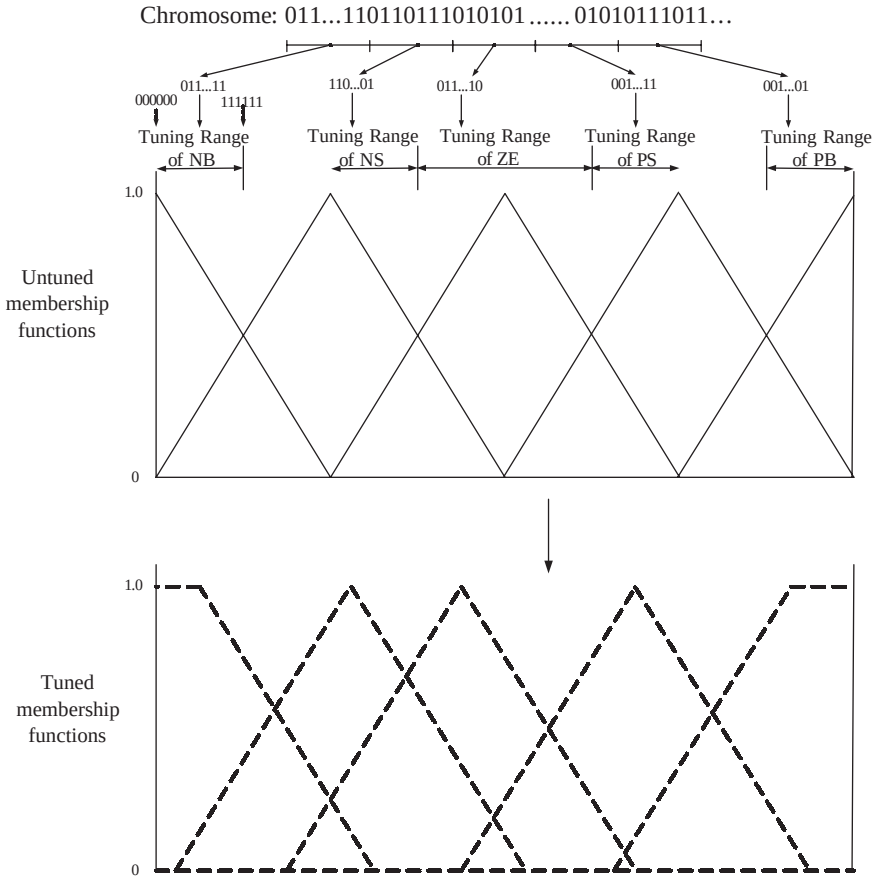


Fig. 7. The encoding method for membership functions of GFLC2 (modified from [10]).

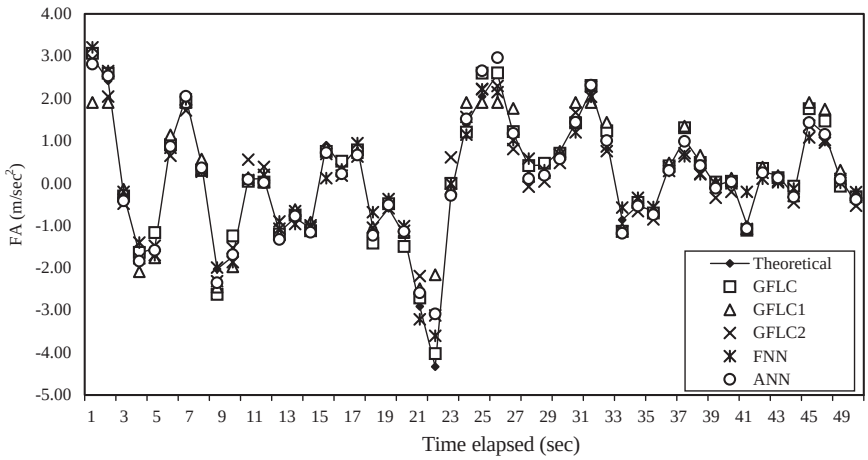


Fig. 8. A comparison of theoretical data and predicted results by different models.

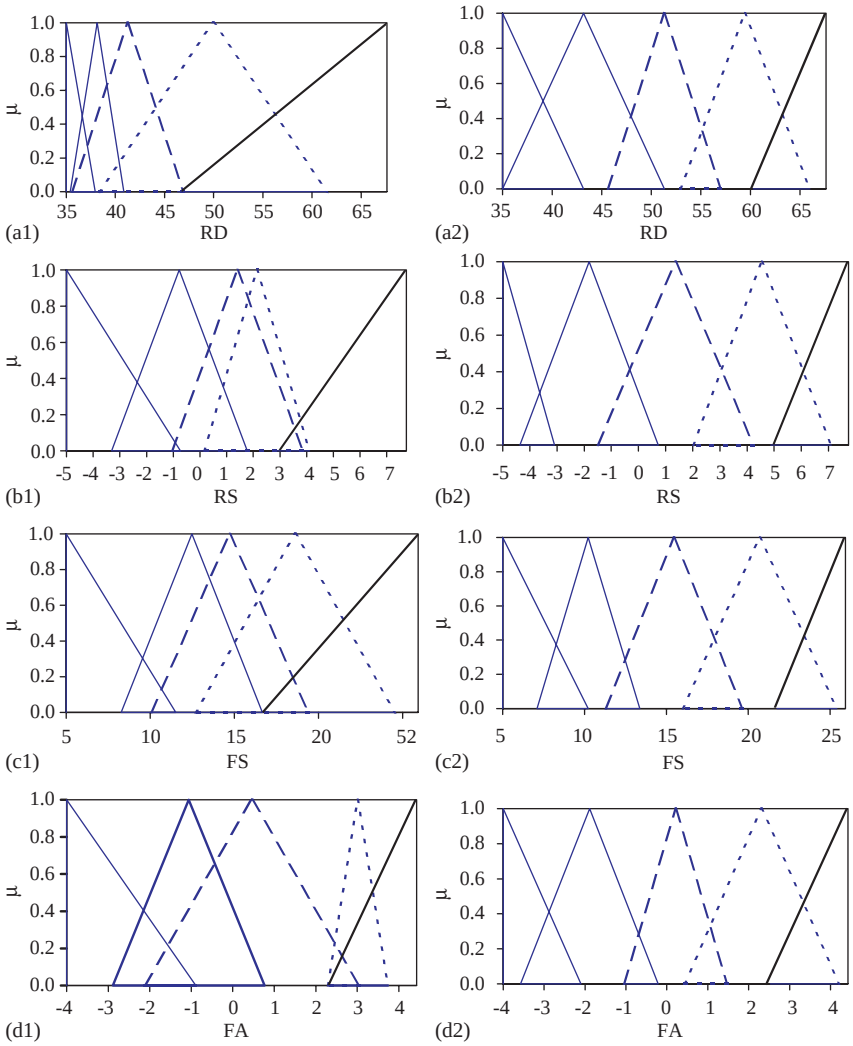


Fig. 9. The tuned membership functions by GFLC, GFLC1, GFLC2 and FNN: (a1) RD by GFLC, (a2) RD by GFLC1; (b1) RS by GFLC, (b2) RS by GFLC1; (c1) FS by GFLC, (c2) FS by GFLC1; (d1) FA by GFLC, (d2) FA by GFLC1; (a3) RD by GFLC2, (a4) RD by FNN; (b3) RS by GFLC2, (b4) RS by FNN; (c3) FS by GFLC2, (c4) FS by FNN; (d3) FA by GFLC2, (d4) FA by FNN.

rules, some of which might be contradictory each other to conclude a neutralized decision. Thus, a further study is worthy of exploring by incorporating a consistency index, proposed by Xiong and Litz [32], into the objective function to avoid the contradiction between rules and to reduce the number of logic rules. Second, some selected rules might be redundant or rarely activated in the training process, inclusion of them or not has little effect on the final decision. Last, the optimal learning results of GFLC might have yet been achieved.

The minimum membership degree with AND connective operator can be the main source of redundant or rarely activated rules. If taking average membership degree instead, the number of selected logic rules

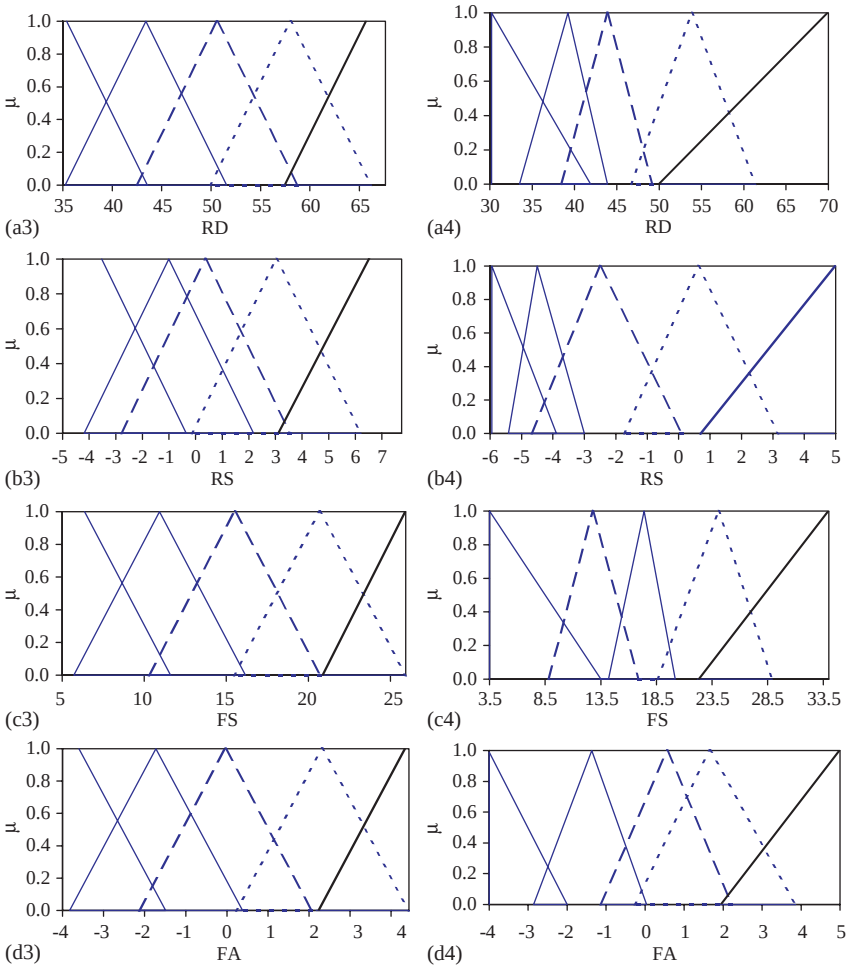


Fig. 9. (*continued*)

can be significantly reduced from 114 to 24. However, the performance of GFLC would become poorer, with values of energy function for training and validating 5.39 and 2.84, respectively.

4.5. The field-observed car-following behaviors

The above analysis is to use the same theoretical car-following data to validate the GFLC, GFLC1, GFLC2, ANN and FNN models and to compare their results. It has concluded that all of these five models can predict the theoretical car-following behaviors rather precisely and that GFLC has outperformed. However, we are not sure if the conclusion is also valid for the real car-following behaviors. To examine this, we further use the field-observed data, composed of 36 s of continuous car-following behaviors obtained by Liu [23], to train the GFLC, GFLC1, GFLC2, ANN and FNN models. The parameter settings of all models are the same as the theoretical case. The predicted accelerations/decelerations of the following

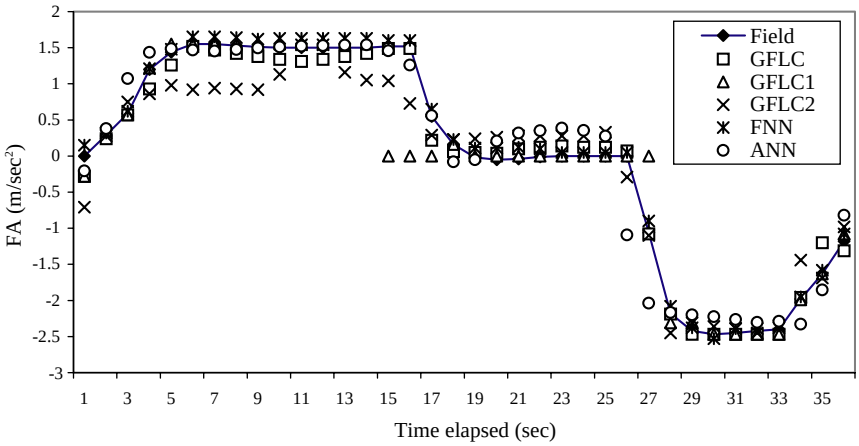


Fig. 10. A comparison of field-observed data and predicted results by different models.

vehicle by these five models are reported in Fig. 10. The values of energy function for GFLC, GFLC1, GFLC2, ANN and FNN are 0.155, 3.105, 2.429, 1.895 and 0.162, respectively. Note that GFLC and FNN models can predict the field-observed car-following behaviors rather precisely and that ANN, GFLC1 and GFLC2 models are not so well performed, especially in the sharp changes in accelerations/decelerations of the following vehicle. In terms of energy function, the performance of GFLC in real application is better than FNN, which is better than ANN and much better than GFLC1 and GFLC2.

5. Concluding remarks

This paper develops a bi-level iterative evolution algorithm to construct a fuzzy logic controller by employing genetic algorithms. A new encoding method is proposed to overcome the problems of too many constraints while tuning the membership functions. The validations from both theoretical and field-observed car-following behaviors have shown the robustness of our proposed GFLC model. It also shows that GFLC performs superior to GFLC1, GFLC2, ANN and FNN. However, the number of logic rules learned by GFLC is still too many to be interpreted as a whole. Even though GFLC has only small improvements over ANN and FNN, GFLC still have some good merits over ANN or FNN. For instance, GFLC can yield both selected logic rules and tuned membership functions while FNN can only obtain tuned membership functions and ANN can yield neither of both. Besides, GFLC can either learn the system from examples or adapt itself to the system without any training data, but ANN and FNN always need a set of training data to learn.

At least four directions for advanced studies on GFLC can be addressed. First, more effective and efficient encoding methods for selecting the logic rules or tuning the membership functions or both deserve to be explored to improve the control performance. Second, it deserves to verify if the learning results of GFLC, the composition of logic rules and the shapes of trained membership functions are interpretable or not. If so, GFLC can be further used to explain an expert's judgment and decision; otherwise it works just like a black box. To deal with the problem of too many selected rules to be interpreted as a whole, a consistency index may be considered in the objective function. Last but not

least, more comparisons can be made among different models in terms of control performance, number of training samples, and interpretations of learning results, etc.

Acknowledgements

The authors would like to acknowledge the constructive comments of two anonymous referees and the research grant from National Science Council of the Republic of China under the contract NSC-90-2416-E-309-008.

References

[1] P.P. Bonissone, P.S. Khedkar, Y. Chen, Genetic algorithms for automated tuning of fuzzy controllers: a transportation application, Proc. Fifth IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'96), 1996, pp. 674–680.
[2] T.C. Chin, X.M. Qi, Genetic algorithms for learning the rule base of fuzzy logic controller, Fuzzy Sets and Systems 97 (1998) 1–7.
[3] I.-F. Chung, C.-J. Lin, C.-T. Lin, A GA-based fuzzy adaptive learning control network, Fuzzy Sets and Systems 112 (2000) 65–84.
[4] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, L. Magdalena, Ten years of genetic fuzzy systems: current framework and new trends, Fuzzy Sets and Systems 141 (2004) 5–31.
[5] O. Cordon, F. Herrera, A three-stage evolutionary process for learning descriptive and approximate fuzzy logic controller knowledge bases from examples, Int. J. Approx. Reason. 17 (4) (1997) 369–407.
[6] O. Cordon, F. Herrera, F. Hoffmann, L. Magdalena, Genetic Fuzzy Systems, World Scientific, Singapore, 2001.
[7] J.M. Del Castillo, F.G. Benitez, On functional form of the speed density relationship: (I) general theory (II) empirical investigation, Transport. Res. 29 (B) (1995) 373–406.
[8] P.Y. Glorennec, Coordination between autonomous robots, Int. J. Approx. Reason. 17 (4) (1997) 433–446.
[9] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[10] H.B. Gurocak, A genetic-algorithm-based method for tuning fuzzy logic controllers, Fuzzy Sets and Systems 108 (1999) 39–47.
[11] F. Herrera, M. Lozano, J.L. Verdegay, Tuning fuzzy logic controllers by genetic algorithms, Int. J. Approx. Reason. 12 (1995) 299–315.
[12] F. Herrera, M. Lozano, J.L. Verdegay, A learning process for fuzzy control rules using genetic algorithms, Fuzzy Sets and Systems 100 (1998) 143–158.
[13] A. Homaifar, E. McCormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, IEEE Trans. Fuzzy Systems 3 (2) (1995) 129–139.
[14] H.-S. Hwang, Control strategy for optimal compromise between trip time and energy consumption in a high-speed railway, IEEE Trans. Systems, Man Cybernet. 28 (6) (1998) 791–802.
[15] R. Jiang, Q.S. Wu, Z.J. Zhu, A new continuum model for traffic flow and numerical tests, Transport. Res. 36 (B) (2002) 405–419.
[16] C.L. Karr, Design of an adaptive fuzzy logic controller using a genetic algorithm, Proc. Fourth Int. Conf. on Genetic Algorithms, 1991, pp. 450–457.
[17] C.L. Karr, Gen. algorithms fuzzy control, AI Expert 6 (2) (1991) 26–33.
[18] C.L. Karr, E.J. Gentry, Fuzzy control of pH using genetic algorithms, IEEE Trans. Fuzzy Systems 1 (1) (1993) 46–53.
[19] J. Kinzel, F. Klawonn, R. Kruse, Modifications of genetic algorithms for designing and optimising fuzzy controllers, Proc. First Int. Conf. on Evolut. Computation, 1994, pp. 28–33.
[20] L.W. Lan, A.Y. Kuo, Y.C. Huang, Color image vehicular detection systems with and without fuzzy neural network: a comparison, J. Chinese Ins. Eng. 26 (5) (2003) 659–670.
[21] A. Lekova, L. Mikhailov, D. Boyadjiev, A. Nabout, Redundant fuzzy rules exclusion by genetic algorithms, Fuzzy Sets and Systems 100 (1998) 235–243.

[22] C.-J. Lin, A GA-based neural fuzzy system for temperature control, Fuzzy Sets and Systems 143 (2004) 311–333.
[23] Y.B. Liu, The study on the design of the indicating system for the adaptive car following and optimal gear shifting, unpublished Ph.D. Dissertation, National Cheng Kung University, Taiwan.
[24] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, Berlin, 1992.
[25] D. Park, A. Kandel, G. Langholz, Genetic-based new fuzzy reasoning models with application to fuzzy control, IEEE Trans. Systems, Man Cybernet. 24 (1) (1994) 39–47.
[26] C.-S. Shieh, Genetic fuzzy control for time-varying delayed uncertain systems with a robust stability safeguard, Appl. Math. Comput. 131 (2002) 39–58.
[27] Y.S. Tarng, Z.M. Yeh, C.Y. Nian, Genetic synthesis of fuzzy logic controllers in turning, Fuzzy Sets and Systems 83 (1996) 301–310.
[28] D. Teodorovic, Fuzzy logic systems for transportation engineering: the state of the art, Transport. Res. 33 (A) (1999) 337–364.
[29] P. Thrift, Fuzzy logic synthesis with genetic algorithms, Proc. Fourth Int. Conf. on Genetic Algorithms, 1991, pp. 509–513.
[30] L.-X. Wang, J. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Systems, Man Cybernet. 22 (1992) 1414–1427.
[31] L. Wang, J. Yen, Extracting fuzzy rules for system modeling using a hybrid of genetic algorithms and Kalman filter, Fuzzy Sets and Systems 101 (1991) 353–362.
[32] N. Xiong, L. Litz, Reduction of fuzzy control rules by means of premise learning—method and case study, Fuzzy Sets and Systems 132 (2002) 217–231.
[33] L. Zadeh, Outline of a new approach to the analysis of complex systems and decision processes, IEEE Trans. Systems, Man Cybernet. SMC-3 (1973) 28–44.