On Generating FC^3 Fuzzy Rule Systems from Data Using Evolution Strategies

Yaochu Jin, *Member, IEEE,* Werner von Seelen, and Bernhard Sendhoff

***Abstract*— Sophisticated fuzzy rule systems are supposed to be flexible, complete, consistent and compact (FC^3). Flexibility, completeness and consistency are essential for fuzzy systems to exhibit an excellent performance and to have a clear physical meaning, while compactness is crucial when the number of the input variables increases. However, the completeness and consistency conditions are often violated if a fuzzy system is generated from data collected from real world applications.**

In an attempt to develop FC^3 fuzzy systems, a systematic design paradigm is proposed using evolution strategies. The structure of the fuzzy rules, which determines the compactness of the fuzzy systems, is evolved along with the parameters of the fuzzy systems. Special attention has been paid to the completeness and consistency of the rule base. The completeness is guaranteed by checking the completeness of the fuzzy partitioning of input variables and the completeness of the rule structure. An index of inconsistency is suggested with the help of a fuzzy similarity measure, which can prevent the algorithm from generating rules that seriously contradict with each other or with the heuristic knowledge. In addition, soft T-norm and BADD defuzzification are introduced and optimized to increase the flexibility of the fuzzy system. The proposed approach is applied to the design of distance controller for cars. It is verified that a FC^3 fuzzy system works very well both for training and test driving situations, especially when the training data are insufficient.

***Index Terms*—Compactness, completeness, consistency, evolution strategies, flexibility, fuzzy rule systems.**

I. Introduction

FUZZY logic has proved to be a very powerful technique in the discipline of system control, especially when the controlled system is hard to be modeled mathematically, or when the controlled system has large uncertainties and strong nonlinearities. Since the last decade, fuzzy control systems have experienced a great success in the fact that not only a lot of successful industrial applications have been found, but concrete theoretical conclusions are also achieved in some important aspects, to name a few, the stability and approximation properties [4], [25], of the fuzzy systems. However, fuzzy systems are by no means perfect. It is criticized that fuzzy control rules are not capable of expressing deep knowledge [6] because they are often established on the basis of experience and intuition of human beings.

With the emergence of the techniques called soft computing [29] or computational intelligence [19], fuzzy control has obtained a new impetus. Backpropagation networks [16], RBF neural networks [9], hybrid pi-sigma networks [11], B-spline networks [8] and neuron-like structures [3] are applied to the adaptation of the fuzzy membership functions and the consequent parameters. On the other hand, counter-propagation networks [30] and BAM [20] are utilized to determine the number of the fuzzy subsets for each input variable. These methods are successful in that it is no longer necessary to determine exactly the parameters of fuzzy rules and the fuzzy partitioning of the input variables in advance.

Genetic algorithms (GA's) have also been employed to design fuzzy systems. Since the first attempt to vary some parameters of a fuzzy rule base using genetic algorithms [13], several efforts have been made to exploit the advantages of GA's for the design of fuzzy systems [18], [21]. It is found that GA's are more flexible because they are capable of optimizing the parameters and the rule number simultaneously [14]. Furthermore, the structure of the fuzzy rules can also be optimized by GA's so that a compact fuzzy rule system can be obtained [10]. One problem that appears in this methodology is the choice of genetic coding. If the conventional coding scheme is used, the length of the chromosome increases significantly with the number of inputs and the number of their fuzzy partitioning. This will no doubt harm the efficiency of the genetic searching. To solve this problem, chromosomes with variable length [5] and context dependent coding [15] have been suggested and proved to exhibit considerable improvements over the fixed length chromosomes.

Relative fewer efforts have been made to date to design fuzzy systems using evolution strategies (ES). We believe that ES are quite suitable for the design of fuzzy systems due to their direct coding scheme and their simple way of handling constraints. In [27], evolution strategies are used to adjust the parameters of the fuzzy rules, and then genetic algorithms are utilized to optimize the structure of the fuzzy rules base. However, since the optimal values of the rule parameters and rule structure depend on each other, it is easy to conceive that it would be better to evolve them simultaneously.

A common problem concerning adjustment of the membership parameters is that the shape of the membership functions is adjusted so drastically that either some of the fuzzy subsets lose their corresponding physical meanings, or the fuzzy subsets do not cover the whole space of the input variable.

Manuscript received August 8, 1998; revised December 24, 1998. This paper was recommended by Associate Editor R. A. Hess.

Y. Jin is with the Industrial Engineering Department, Rutgers, The State University of New Jersey, Piscataway, NJ 08854 USA. He is also with the Electrical Engineering Department, Zhejiang University, Hangzhou 310027, China.

W. von Seelen and B. Sendhoff are with Institut für Neuroinformatik, Ruhr-Universität Bochum, D-44780 Bochum, Germany.

Publisher Item Identifier S 1083-4419(99)05274-7.

1083–4419/99$10.00 © 1999 IEEE

In the latter case, the fuzzy partitioning is called incomplete, i.e., the fuzzy system takes no action if the value of the variable falls in the uncovered region. Besides, no sufficient research work has been carried out to keep the consistency of the fuzzy rules in generating fuzzy rules from data. In most cases, only the rules that have the same antecedent but different consequent are considered to be inconsistent. In [26], a degree of belief is assigned to each generated rule and only the one with a maximal degree will be accepted if two rules have the same IF part but different THEN parts. In [10], the priority is given to the rule that first appears. As a matter of fact, rules that have different IF parts might also be inconsistent, either with other rules or with the human heuristics.

To cope with the above mentioned problems, an ES based methodology for generating fuzzy systems is proposed in this paper. It focuses mainly on the completeness, consistency and compactness of the fuzzy systems. Another feature of this work is that the fuzzy operators, including T-norms and BADD defuzzification, are also optimized. It is demonstrated with an example of distance control that the proposed approach is advantageous over the other methods in the following respects. 1) The fuzzy system is compact and efficient because the number of the fuzzy rules is greatly reduced; 2) The fuzzy system is complete and no seriously conflicting rules will be generated, which contributes to the improvement of the generalization ability of the fuzzy system and guarantees that the knowledge acquired by the fuzzy rules is physically sound, and 3) The fuzzy system is expected to exhibit a better flexibility because soft fuzzy operators [28] are incorporated and optimized.

In the next section, the definition of FC^3 fuzzy systems, including the concept of soft T-norm, BADD defuzzification, completeness, consistency and compactness are provided. In Section III, design of an FC^3 fuzzy system using evolution strategies is given in detail. An application example of distance control is described in Section IV. Various simulations are carried out to show that an FC^3 fuzzy system exhibits excellent training and test performances even if the training data are insufficient. Finally, a summary of the paper is given in Section V.

II. FC^3 Fuzzy Systems

A. Basic Formulas of Fuzzy Systems

Until now, two main types of fuzzy systems, namely, the Mamdani type and the Takagi–Sugeno type, have been developed. The main difference between these two types of fuzzy systems lies in the fact that the consequent part of Takagi–Sugeno rules are concrete values instead of fuzzy sets. Since a multi-input multi-output fuzzy system can always be separated into a group of multi-input single-output (MISO) fuzzy systems, we discuss here only the MISO fuzzy systems without the loss of generality. For an MISO fuzzy system with n input variables, the Mamdani type fuzzy rules are expressed in the following form:

$$R_i\text{: If } x_1 \text{ is } A_{i1} \text{ and } x_2 \text{ is } A_{i2} \text{ and } \cdots \text{ and } x_n \text{ is } A_{\text{in}}, \quad \text{then } y \text{ is } B_i; \qquad i = 1, 2, \cdots, N \tag{1}$$

where

x_j $(j = 1, 2, \cdots, n)$	input variables;
y	output of the fuzzy system;
A_{ij} and B_i	linguistic terms defined by corresponding membership functions $A_{ij}(x_j)$ and $B_i(y)$.

For a conventional fuzzy system, if each variable is divided into M fuzzy subsets, then the total rule number is $N = M^n$. It is noticed that for such a fuzzy system, the number of rules increases exponentially with the number of input variables. According to the Mamdani fuzzy implication method, the fuzzy relation of the ith rule can be expressed by

$$R_i = (A_{i1} \times A_{i2} \times \cdots \times A_{in}) \times B_i \tag{2}$$

which is a fuzzy set whose membership function is described by

$$\begin{aligned} &R_i(x_1, x_2, \cdots, x_n, y) \\ &\quad = A_{i1}(x_1) \mathcal{T} A_{i2}(x_2) \mathcal{T} \cdots \mathcal{T} A_{\text{in}}(x_n) \mathcal{T} B_i(y) \end{aligned} \tag{3}$$

where $\mathcal{T}$ means the T-norm operator. Based on sup-star composition, the overall fuzzy relation of the fuzzy system in terms of membership function can be written as follows:

$$R(x_1, x_2, \cdots, x_n, y) = \bigvee_{i=1}^{N} R_i(x_1, x_2, \cdots, x_n, y) \tag{4}$$

where $\vee$ is the maximum operator. Suppose B_i is normal, i.e., the maximum value of $B_i(y)$ is 1.0, where y_i is the point at which $B_i(y)$ reaches its maximum, and the center of gravity (COG) defuzzification method is adopted, then the crisp output of the Mamdani fuzzy system is obtained by

$$y = \frac{\sum_{i=1}^{N} \{\mathcal{T}_{j=1}^{n} A_{ij}(x_j) \cdot y_i\}}{\sum_{i=1}^{N} \mathcal{T}_{j=1}^{n} A_{ij}(x_j)}. \tag{5}$$

If Takagi–Sugeno type fuzzy rules are used, then the fuzzy rules have the following form:

$$R_i\text{: If } x_1 \text{ is } A_{i1} \text{and } x_2 \text{ is } A_{i2} \text{ and } \cdots \text{ and } x_n \text{ is } A_{\text{in}}, \quad \text{then } y_i = f_i(x_1, x_2, \cdots, x_n) \qquad i = 1, 2, \cdots, N. \tag{6}$$

It is observed that the consequent part of the Takagi–Sugeno rules is a crisp value and usually a function of the input variables instead of a linguistic variable. The output of a Takagi–Sugeno rule system is in the following form:

$$y = \frac{\sum_{i=1}^{N} \{\mathcal{T}_{j=1}^{n} A_{ij}(x_j) \cdot f_i(x_1, x_2, \cdots, x_n)\}}{\sum_{i=1}^{N} \mathcal{T}_{j=1}^{n} A_{ij}(x_j)} \tag{7}$$

If $f(\cdot)$ is a linear function in the form of

$$f_i(x_1, x_2, \cdots x_n) = p_{i0} + p_{i1}x_1 + p_{i2}x_2 + \cdots + p_{in}x_n \tag{8}$$

where p_{ij} $(i = 0, 1, 2, \cdots, N; j = 1, 2, \cdots, n)$ are constant coefficients, then the rule structure can be determined with a search algorithm and the parameters of the rules can be identified using the least-square method or gradient method [24]. If a more complex combination of the input variable is needed, then it can be handled with a pi–sigma neural network [11].

In practice, the consequent part of the Takagi–Sugeno rules is often simplified to a constant, in which case the output of the Takagi–Sugeno rules can be written as follows:

$$y = \frac{\sum_{i=1}^{N} \{\mathcal{T}_{j=1}^{n} A_{ij}(x_j) p_{i0}\}}{\sum_{i=1}^{N} \mathcal{T}_{j=1}^{n} A_{ij}(x_j)}. \tag{9}$$

From (5) and (9), it is easy to notice that the same conclusion is derived from Mamdani rules and Takagi–Sugeno rules.

In the following subsections, the definition of an FC^3 fuzzy system in the context of this paper will be provided in detail and the reasons to construct such fuzzy systems are given.

B. Flexible Fuzzy Operators

In the above fuzzy systems, the fuzzy operators, including T-norm and defuzzification method, are conventional T-norms and COG defuzzifier. Although several T-norms have been proposed, they have not been shown to play an important role in improving the performance of the fuzzy control systems [7], [12]. On the other hand, which defuzzification method should be selected is in fact problem-related [22]. In these respects, the so-called soft T-norm and BADD defuzzifier are very flexible [28]. They can be expressed in the following form:

$$\text{Soft T-norm:} \quad \tilde{\mathcal{T}}(x_1, x_2, \cdots, x_n) = (1-\alpha)\frac{1}{n}\sum_{i=1}^{n} x_i + \alpha \mathcal{T}(x_1, x_2, \cdots, x_n) \tag{10}$$

$$\text{BADD defuzzifier:} \quad y = \frac{\sum_{i=1}^{N} \{[\mathcal{T}_{j=1}^{n} A_{ij}(x_j)]^{\delta} p_{i0}\}}{\sum_{i=1}^{N} [\mathcal{T}_{j=1}^{n} A_{ij}(x_j)]^{\delta}} \tag{11}$$

where $\alpha \in [0, 1]$ and $\delta \in [0, \infty)$. It is noticed that if α is 1.0, then the soft T-norm reduces to the conventional T-norm. If α is 0, then the soft T-norm is equivalent to computing a mean value. It is argued in [12] that such a soft T-norm is more promising in improving the performances of the fuzzy systems than the combination of different T-norms. On the other hand, the BADD defuzzifier is equivalent to the center of gravity (COG) method and the mean of maximum (MOM) method when $\delta = 1$ and $\delta \to \infty$, respectively. If δ varies from 1 to ∞, BADD defuzzifier is able to reach a possible compromise between COG and MOM defuzzifiers. In this way, the hard task to make a choice among the different fuzzy operators can be spared and the fuzzy system is flexible.

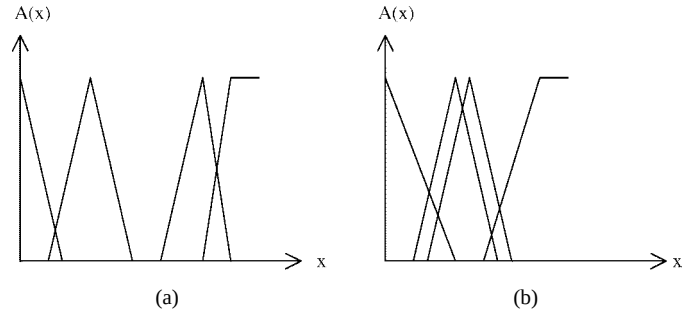


Fig. 1. Overfitting of the membership functions. (a) Incomplete fuzzy partitioning and (b) lack of distinguishability.

C. Completeness of the Fuzzy Systems

The discussion of completeness is necessary if a fuzzy system is generated automatically from data. In order to discuss the completeness of the fuzzy system, it is desirable to provide a definition of the completeness. In this paper, a fuzzy system is said to be complete if

1) fuzzy partitioning of each input variable is complete;
2) rule structure of the fuzzy system is complete.

If one of the above conditions is violated, the fuzzy rule system is incomplete, which implies that the fuzzy system will provide no output in some cases. Although it is suggested to output a value of zero or some other values in case no rules are fired, it will be shown in this work that incompleteness of the fuzzy systems should be avoided.

We first discuss the completeness of the fuzzy partitionings of the input variables. Suppose input variable x is partitioned into M fuzzy subspaces represented by $A_1(x), A_2(x), \cdots, A_M(x)$ on the universe of discourse U, then the partitioning is considered to be complete if the following condition holds:

$$\forall_{x \in U} \exists_{1 \le i \le M} A_i(x) > 0. \tag{12}$$

It is often the case that the fuzzy partitionings of some input variables are no longer complete after the fuzzy membership functions have been optimized. This is quite easy to understand because all the optimization algorithms try to adjust the distribution of the membership functions according to the distribution of the given data. Such optimizations, however, will give rise to the problem of "overfitting" of the fuzzy membership functions, when the presented data distribute irregularly. The overfitting of the fuzzy membership functions results in the following consequences:

1) fuzzy partitionings become incomplete;
2) physical meaning of some fuzzy subsets may be blurred, that is to say, the fuzzy subsets lack distinguishability (see Fig. 1).

If the membership functions are realized, say, by neural networks, they may become neither unimodal nor normal. All these phenomena might be beneficial to improve the training performance, but will usually deteriorate the generalization quality of the fuzzy systems.

In order to avoid overfitting of the membership functions, certain measures must be taken in the process of parameter optimization or rule generation. One practical measure is

to limit the adjustable range of the parameters so that the completeness of the fuzzy partitioning will be kept and the distinguishability of different fuzzy sets will be preserved. Another method is to add or merge some fuzzy subsets in the process of optimization if needed. In this paper, we suggest a new possibility to deal with these problems with the help of fuzzy similarity measures.

A fuzzy similarity measure indicates the degree to which two fuzzy sets are equal. It has been used in structure learning of fuzzy systems [17]. In their work, the fuzzy similarity measure is used to add new membership functions for the output variable so that proper fuzzy partitioning of the output space can be obtained. In our approach, the fuzzy similarity measure is used to preserve the completeness of the fuzzy partitionings of the input variables and to preserve the distinguishability of the fuzzy subsets. For any two fuzzy sets A and B, the fuzzy similarity measure is defined by:

$$S(A,B) = \frac{M(A \cap B)}{M(A \cup B)} = \frac{M(A \cap B)}{M(A) + M(B) - M(A \cap B)} \tag{13}$$

where $M(A)$ is called the size of fuzzy set A and can be calculated as follows:

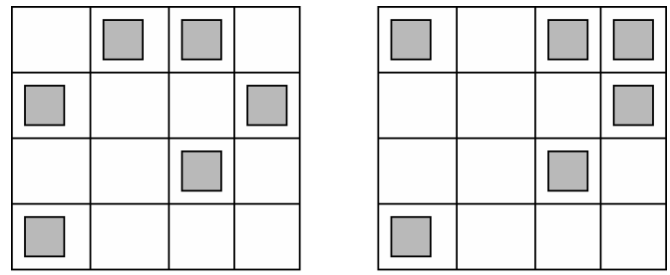
$$M(A) = \int_{-\infty}^{+\infty} A(x)\, dx \tag{14}$$

It is noticed that $S(A,B) = 1$ if and only if $A = B$ and $S(A,B) = 0$ if and only if A and B do not overlap. In other cases, $S(A,B)$ varies from 0 to 1. Therefore, if the fuzzy similarity measure of any two neighboring fuzzy sets is controlled properly, the incompleteness of the fuzzy partitioning can be avoided and the distinguishability of the fuzzy sets can be preserved. Calculation of the fuzzy similarity measure in the case of triangular membership functions and Gaussian membership functions are provided in Appendix I.

It should be pointed out that the completeness of the fuzzy partitionings does not necessarily guarantee the completeness of the fuzzy systems, in other words, a fuzzy rule system may still be incomplete even if the fuzzy partitionings of the input variables are complete. This happens when the rule structure is incomplete, i.e., some of the fuzzy subsets are not used by the rule system, which is often the case in the course of rule structure optimization (see Fig. 2). The completeness of the rule structure will be explained further when we discuss the compactness of the fuzzy systems.

D. Consistency of the Fuzzy Systems

The problem of consistency of the fuzzy rules is usually thought to be trivial if the rules are extracted from expert knowledge. However, if the rules are automatically generated from a set of data affected by noise, this problem can become serious. We discuss here not only the consistency among the fuzzy rules in the generated rule base, but also the consistency



(a) (b)

Fig. 2. Rule structures of a two dimensional fuzzy system. A shaded area represents a fuzzy rule. (a) Complete rule structure. (b) Incomplete rule structure.

of the rules with the intuition and common sense of human beings. Therefore, fuzzy rules are regarded as inconsistent, if

- They have *very similar* premise parts, but possess *rather different consequents*, and
- They conflict with the expert knowledge or heuristics.

It is noticed that the concept of consistency is not concrete and can only be described by a value of degree. Moreover, the discussion of consistency is sensible only if the premise parts of the rules are very similar, if not necessarily the same. That is to say, two fuzzy rules may contradict with each other even if they do not have the same premise, on the other hand, it is hard to say that two rules are inconsistent if their premise parts have little similarity. Suppose there are two rules in the following form:

R_i: *If error is PS and change of error is ZO, then change of control is PB*

R_k: *If error is ZO and change of error is ZO, then change of control is NB*

where "NB," "ZO," "PS," and "PB" represent negative big, zero, positive small and positive big respectively. Despite that the antecedents of the two rules are not the same, they seem to be inconsistent because they have very similar premises,[1] but rather different consequents. However, if the two "ZO's" in rule k are replaced by "NB," then these two rules are quite consistent even if their consequents are completely different.

Therefore, it is necessary to provide a proper definition of consistency, which can embrace the aforementioned considerations. Before we discuss the definition of the consistency, we first provide the definition of *the similarity of rule premise* (SRP) and *the similarity of rule consequent* (SRC) again with the help of fuzzy similarity measure. Consider two rules in the rule base:

R_i: If x_1 is $A_{i1}(x_1)$ and x_2 is $A_{i2}(x_2)$ and $\cdots$ and x_n is $A_{\text{in}}(x_n)$, then y is $B_i(y)$

R_k: If x_1 is $A_{k1}(x_1)$ and x_2 is $A_{k2}(x_2)$ and $\cdots$ and x_n is $A_{kn}(x_n)$, then y is $B_k(y)$

Then SRP and SRC of these two rules are defined as follows:

$$\text{SRP}(i,k) = \min_{j=1}^{n} S(A_{ij}, A_{kj}) \tag{15}$$

[1] Of course, it depends on the definition of the membership functions.

$$\mathrm{SRC}(i,k) = S(B_i, B_k) \tag{16}$$

where n is the total number of the input variables and $S(A,B)$ is the fuzzy similarity measure of fuzzy sets A and B as defined in (13). Then the consistency of rule $R(i)$ and $R(k)$ is defined by:

$$\mathrm{Cons}(R(i),R(k)) = \exp\left\{-\frac{\left(\dfrac{\mathrm{SRP}(i,k)}{\mathrm{SRC}(i,k)} - 1.0\right)^2}{\left(\dfrac{1}{\mathrm{SRP}(i,k)}\right)^2}\right\} \tag{17}$$

The above definition of consistency has two fundamental characteristics. One characteristic is that the degree of consistency tends to be high when the SRP and SRC of two rules is in proportion, provided that the SRP of the two rules is high. Particularly, if the rules have the same premise and the same consequent, the degree of consistency reaches its highest value of 1. If the premises are the same but the consequents are different, then the consistency ranges from 0 to 1.0. The other characteristic is that the degree of consistency is always high if the SRP of two rules is very low, no matter how the relation of SRP and SRC changes. This is concordant with the assumption that two rules will always be considered to be consistent if they have very different premises. It can be seen that our definition of consistency is a soft criterion, which is in good agreement with the philosophy of fuzzy set theory. Nevertheless, this definition is constructed from the point of application and has to be subject to strict theoretical examinations.

One further fact that needs to be pointed out is that the consistency definition is generally suitable for Mamdani fuzzy systems. If the consequent part of the Takagi–Sugeno fuzzy systems is reduced to a constant, then it is also applicable because an output in the form of fuzzy singleton can always be extended to a normal fuzzy set. However, if the consequent part of the Takagi–Sugeno rule is a function of the input variables, it is generally difficult to evaluate the consistency of two fuzzy rules, because a consequent variable expressed in terms of a real function does not exhibit clear physical meanings.

Apart from the consistency checking among the rules, it is also important to investigate the consistency of the generated rules with the human intuition or prior knowledge. If the human intuition or prior knowledge is expressed in fuzzy rules, then the consistency checking can also be implemented using the definition provided above. The prior knowledge required here is fundamental, which can be normally obtained from common sense and intuition. Take driving situations as an example, the following rule is easily available:

If **the distance** is *very small* and **the speed** is *very high*, then **the acceleration** should be *negative*.

Although such rules are very simple, they are critical to the performance of fuzzy systems. Despite that fuzzy systems are believed to be able to tolerate some inconsistent rules to a certain degree, rules that seriously contradict with the others or with the human heuristics definitely lower the performance of the fuzzy systems.

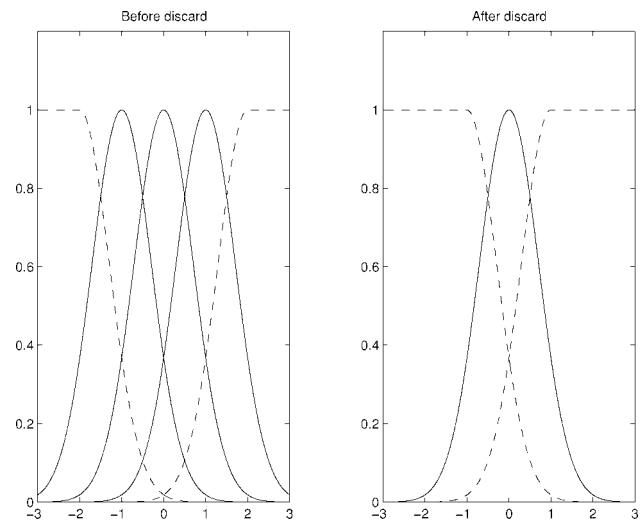


Fig. 3. Fuzzy sets at the two ends of the partition (described with dotted lines) can be discarded. (a) Before discard. (b) After discard.

E. Compactness of the Fuzzy Systems

The number of fuzzy rules needed to represent a physical system also depends on the structure of the fuzzy rules. If a fuzzy system has n inputs and each input variable is partitioned into M subspaces, there will be M^n rules in a fuzzy system with a standard structure. The standard structure is usually not optimal and therefore not compact [10]. A compact fuzzy system is very desirable when the number of input variables increases, especially for Takagi–Sugeno fuzzy systems with general consequent forms.

Some measures should be taken to guarantee the completeness of the fuzzy rule system in optimizing the rule structure. If some of the fuzzy subsets are not used, the rule structure will be incomplete. However, it is against our aim if we require that all fuzzy subsets must be used by the fuzzy system. A way out of this dilemma is that we allow the fuzzy system to discard the fuzzy subsets at the two ends of the fuzzy partitioning; however, a subset inside the fuzzy partitioning must be used (see Fig. 3).

III. Generation of FC^3 Fuzzy Systems Based on Evolution Strategies

A. Algorithm of Evolution Strategies

Evolution strategies (ES) are used to optimize the coefficients of the soft T-norm and BADD defuzzifier, the parameters of the fuzzy membership functions as well as the structure of the fuzzy rules. Evolution strategies, instead of genetic algorithms are used in this paper due to the following two considerations. One is that the coding of ES is direct for real valued parameter optimization and consequently the length of the string increases just linearly with the number of variables. Comparative studies on ES's and GA's have been carried out in [2] and better results have been obtained by using ES's in the case of real valued parameter optimization. Because the rule structure will be evolved together with the parameters of the membership functions, the number of variables to be optimized increases significantly. If a binary

genetic algorithm is used, the length of the string will grow drastically, which no doubt affects the searching efficiency seriously. To alleviate such difficulties, either the range of the parameters needs to be limited, or special coding methods should be developed. The other reason is that ES deals with the constraints quite conveniently, which enables the algorithm to search a wider parameter space. However, this does not imply that ES is superior to GA, or vice versa.

Presently, several ES algorithms are available. The two most widely used algorithms are noted as $(\mu+\lambda)$-ES and (μ,λ)-ES. The former selects the best μ individuals from both of the μ parents and λ offsprings as the parents of the next generation, while the latter selects the best μ parents only from the λ offsprings. It is believed that the (μ,λ)-ES outperforms the $(\mu+\lambda)$-ES because (μ,λ)-ES is less likely to get stuck in local optima. The numbers of parents and offsprings are recommended to be at a ratio of $\mu/\lambda \approx 1/7$ [23].

Since both real and integer numbers are involved in the optimization, a slightly modified version of (μ,λ)-ES [1] is used here. An ES algorithm that is capable of dealing with mixed optimization can be described by the following notation:

$$(\mu,\lambda)\text{-ES} = (I,\mu,\lambda;m,s,\sigma;f,g) \tag{18}$$

where I is a string of real or integer numbers representing an individual in the population, μ and λ are the numbers of the parents and offsprings respectively; σ is the parameter to control the step size, m represents the mutation operator, which is the main operator in the mechanism of ES. In ES algorithms, not only the variables, but also the step-size control parameter σ are mutated. In (18), parameter s stands for the selection method and in this case, the parents will be selected only from the λ descendants; f is the objective function to be minimized, and g is the constraining function to which the variables are subject. The variables to be optimized and the step-size control parameter are mutated in the following way:

$$\sigma_i' = \sigma_i \cdot \exp\left(\tau_1 \cdot N(0,1) + \tau_2 \cdot N_i(0,1)\right), \quad i = 1,2,\cdots,Q \tag{19}$$

$$I_i' = I_i + \sigma_i' \cdot N_i(0,1), \qquad i = 1,2,\cdots,Q_1 \tag{20}$$

$$I_i' = I_i + \lfloor \sigma_i' \cdot N_i(0,1) \rfloor, \quad i = Q_1+1, Q_1+2,\cdots,Q \tag{21}$$

where $N(0,1)$ and $N_i(0,1)$ are normally distributed random numbers with mean of zero and variance of 1, Q is the total number of variables to be optimized, Q_1 is the number of real variables and naturally $Q-Q_1$ denotes the number of the integer variables, and "$\lfloor x \rfloor$" is the maximal integer smaller than x. In our case, the parameters representing the fuzzy operators and membership functions are encoded with real variables, while the rule structure parameters are encoded with integer numbers. τ_1 and τ_2 are two global step control parameters.

Similar to genetic algorithms, a lot of strategy parameters of ES, which have great influence on the performance of the algorithm, must be fixed manually. These include the population size μ and λ, the global step control parameters τ_1 and τ_2, and the initial values of step-size σ. The optimization problems in the real world normally have a lot of suboptima, in which a standard evolution strategy can get trapped. To acquire a solution as good as possible, it is desired to improve the performance of the standard ES. In this paper, a minor modification is made and, nevertheless, is proved to be effective. In practice, we find that the process of evolution stagnates when the step-size control σ converges to zero prematurely. To prevent the step-size from converging to zero, we re-initialize it with a value of, say, 1.0, when σ becomes very small. This enables the algorithm to escape from the local minima on the one hand, on the other hand, gives rise to some oscillations of the performance. Therefore, it is important to record the best individual which has been found so far. However, this best individual does not take part in the competition of selection if it does not belong to the current generation.

B. Coding of the Fuzzy Rule Parameters

The genetic coding of the variables for the fuzzy operators and fuzzy membership functions is very direct. Only the constraints of the variables need to be addressed. In our case, the soft coefficient for T-norm covers the range between 0 and 1.0. Theoretically, the coefficient of BADD defuzzification can vary from 0 to infinity, however, an upper bound is imposed in our implementation. Without loss of generality, the following Gaussian membership functions are used:

$$A(x) = \exp\left\{-\frac{(x-c)^2}{w^2}\right\}. \tag{22}$$

Therefore, each membership function has two parameters, namely, center c and width w. In order that all the subsets of a fuzzy partitioning can distribute as freely as possible provided that the completeness condition is satisfied, the centers of each fuzzy membership function can move on the universe of discourse of the corresponding variable, which is limited by the physical system. Of course, all the fuzzy subsets should be ordered according to their centers so that the checking of completeness can be done and that the mechanism of the rule structure optimization works properly. As for the widths of the membership functions, they are loosely limited so that they are greater than zero and naturally, not wider than the whole space. In fact, they will be subject to the completeness conditions and the distinguishability requirements.

C. Coding of the Rule Structure

The rule structure coding is important because the size of a fuzzy system is fully specified by the rule structure. Suppose each input variable x_i has a maximal number of fuzzy subsets M_i, then the rule base has at most $N = M_1 \times M_2 \times \cdots \times M_n$ fuzzy rules if there are n input variables. Thus, the premise structure of the rule system can be encoded by the following matrix:

$$\text{Struc}_{\text{premise}} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{N1} & a_{N2} & \cdots & a_{Nn} \end{bmatrix} \tag{23}$$

where $a_{ji} \in \{0,1,2,\cdots,M_i\}, j = 1,2,\cdots,N; i = 1,2,\cdots,n$. The integer numbers of $1,2,\cdots,M_i$ represent the corresponding fuzzy subsets of x_i, while $a_{ji}=0$ indicates that variable x_i does not appear in the jth rule. It is argued

that the assumption of the largest number of fuzzy partitioning will not harm the compactness of the rule system, because the redundant subsets will be discarded automatically by the algorithm due to the checking of distinguishability. Similarly, the structure of the consequents can be encoded with a vector of positive integers:

$$\mathrm{Struc}_{\mathrm{consequent}} = [c_1, c_2, \cdots, c_N]^T \tag{24}$$

where $c_j \in \{1, 2, \cdots, K\}, j = 1, 2, \cdots, N$, supposing that the consequent variable has at most K fuzzy subsets. This works both for Mamdani type rules and Takagi–Sugeno fuzzy rules whose consequents are constants. If the Takagi–Sugeno rules have a real function of input variables as consequents, the coding of the consequent structure is not necessary.

D. Completeness and Consistency Checking

The completeness checking consists of the fuzzy partitioning checking and the rule structure checking. At first, the completeness of the fuzzy partitioning of each input variable is examined using the fuzzy similarity measure. One advantage of using fuzzy similarity measure (FSM) is that by regulating the grade of FSM, the degree of the overlapping of two subsets can be properly controlled. If FSM of two neighboring fuzzy subsets is zero or too small, it indicates that either the fuzzy partitioning is incomplete or they do not overlap enough. On the other hand, if FSM is too big, then it means that the two fuzzy subsets overlap too much and the distinguishability between them is lost. To keep the fuzzy membership functions in a proper shape, the fuzzy similarity measure of any two neighboring membership functions is required to satisfy the following condition:

$$\mathrm{FSM}^- \leq S(A_i, A_{i+1}) \leq \mathrm{FSM}^+ \tag{25}$$

where A_i and A_{i+1} are two neighboring fuzzy sets, FSM^- and FSM^+ are the desired lower and upper bound of the fuzzy similarity measure, respectively. If this condition is not satisfied, the fitness index of the generated fuzzy system will be assigned to a very large value so that the corresponding individual can hardly survive. In our case the optimization task is minimization, therefore a lower fitness value is better. The label "fitness" here simply represents the cost function and has nothing to do with its biological notation.

Although a consistency index is given in Section II, it can not be directly applied in our evolutionary algorithms. To solve this problem, a degree of inconsistency of a rule base is suggested based on the consistency index provided in the last section. At first, a degree of *inconsistency* for the ith rule is calculated as follows:

$$\begin{aligned}\mathrm{Incons}(i) = &\sum_{\substack{1\leq k\leq N\\ k\neq i}} [1.0 - \mathrm{Cons}(R^1(i), R^1(k))] \\ &+ \sum_{1\leq l\leq L} [1.0 - \mathrm{Cons}(R^1(i), R^2(l))]; \\ &i = 1, 2, \cdots, N\end{aligned} \tag{26}$$

where R^1 and R^2 denote the rule base generated from data and the rule base extracted from prior knowledge, N and L are the rule numbers of R^1 and R^2, respectively. The degree of inconsistency of each rule is then summed up to indicate the degree of inconsistency of a rule base:

$$f_{\mathrm{Incons}} = \sum_{i=1}^{N} \mathrm{Incons}(i) \tag{27}$$

which can be incorporated in the objective function of the evolutionary algorithm.

Note that no special measures are taken to reduce the number of the fuzzy rules. In practice, it is found that the evolutionary algorithm tends to select a more compact system rather than a standard system. This implies that a standard fuzzy system normally has a worse performance than a compact system.

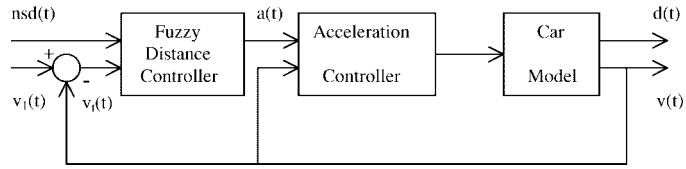


Fig. 4. Control diagram of the driving system.

IV. Application Example

Fuzzy systems play a unique role in control systems where a human controller is an essential part. This is true for driving a car. A skilled driver can control a car successfully according to the situation he is in. However, it is almost impossible to describe the driving behavior using an exact mathematical model because not only physical variables but also personal experience and habits are involved. In this section, we try to generate a fuzzy rule system to simulate the driving behavior based on the collected data using the proposed method.

A. Description of the Distance Control System

A diagram of the distance control system is illustrated in Fig. 4, where $v(t)$ and $v_1(t)$ are the velocities of the controlled car and of the car in front of it, $d(t)$ is the distance between the two cars, $v_r(t) = v_1(t) - v(t)$ denotes the relative speed, and $nsd(t)$ is the normalized safety distance, which is calculated as follows:

$$nsd(t) = \frac{d(t) - s(t)}{s(t)} \tag{28}$$

where $s(t)$ is called the safety distance. It is found that $s(t)$ is basically in proportion to the speed $v(t)$, however, it seriously depends on the driver. Not only different drivers have different views on safety distance, but the same driver also makes various decisions. Our task is to design a fuzzy distance controller that is able to produce a correct acceleration based on the data collected from different drivers in different driving situations. In this paper, data from five driving situations are used.

B. Simulations and Discussions

Before we carry out the simulations, we first discuss the selection of the fitness function so that correct fuzzy rules can

TABLE I
RULE BASE (WITHOUT CHECKING, SITUATION 1)

	0	1	2	3	4	5
0	*	*	*	3	*	*
1	*	2	3	*	2	6
2	*	2	3	2	4	5
3	*	2	2	2	*	6
4	*	2	3	6	7	7
5	*	*	3	6	7	*

TABLE II
INCONSISTENCY INDEXES (WITHOUT CHECKING, SITUATION 1)

	0	1	2	3	4	5
0	*	*	*	0.00	*	*
1	*	0.24	0.37	*	1.37	0.096
2	*	0.22	0.54	1.00	1.17	0.53
3	*	0.20	0.87	1.03	*	0.27
4	*	0.27	0.21	1.99	1.99	0.49
5	*	*	0.20	0.20	0.10	*

be generated. Three candidates, namely, speed, acceleration and normalized distance, can be used as an index to evaluate the performance of the fuzzy controller. However, it is found in practice that only the speed index is able to evaluate the fuzzy rules effectively. Therefore, the following objective function is adopted:

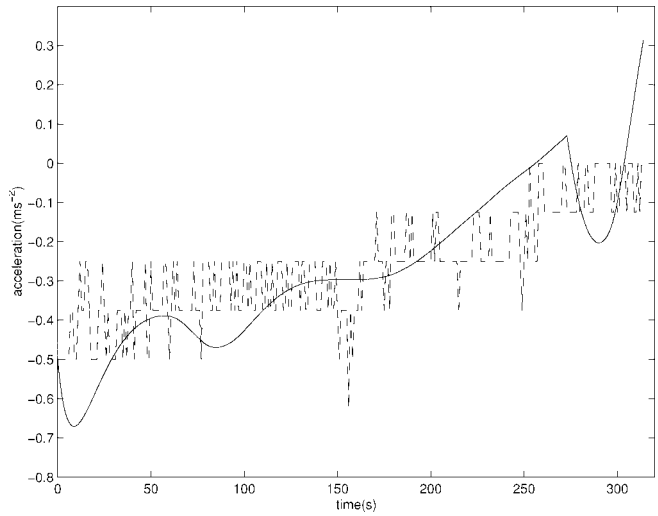
$$f_E = \sum_{t=1}^{J} \sqrt{(v(t) - v^d(t))^2} \tag{29}$$

where J is the total number of sampled data and v_d is the target velocity. Combining the completeness and inconsistency indices, the quality of a generated fuzzy rule system is evaluated with the following objective function:

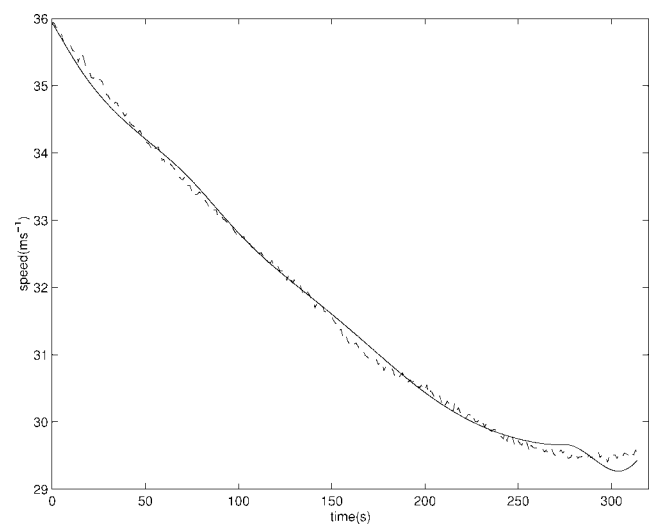
$$f = f_E + \xi \cdot f_{\text{Incons}} + f_{\text{Incompl}} \tag{30}$$

where f_E and f_{Inconsis} are provided in (27) and (29), respectively, f_{Incompl} is a penalty term for the rule system if the completeness condition in (25) is not satisfied or the rule structure is incomplete; ξ is a weighting constant to control the consistency level. In general, once the rule system is found to be incomplete, the penalty term f_{Incompl} is so large that the individual is not able to survive. That is to say, the evolutionary algorithm tolerates some degree of inconsistency, but allows no incomplete fuzzy systems.

Based on the collected data, the meaningful range of normalized safety distance $nsd(t)$, relative speed $v_r(t)$ and the acceleration $a(t)$ are selected as $[-1, 5]$, $[-10, 10]$, and $[-3, 3]$, respectively. As mentioned before, the centers of all the fuzzy membership functions are allowed to vary over the whole space of the corresponding variables. We suppose both nsd and v_r have a maximum of five fuzzy subspaces and a has maximally eight fuzzy subspaces. Therefore, the fuzzy system has at most 25 fuzzy rules.



(a)



(b)

Fig. 5. Training results (without checking, situation 1). (a) Acceleration and (b) speed.

The following two rules are used as the prior knowledge in our research.

If nsd is *Positive Big* and v_r is *Positive Big*,
then a is *Positive Big*
If nsd is *Negative Big* and v_r is *Negative Big*,
then a is *Negative Big*

No doubt, such prior knowledge is quite straightforward and is easy to obtain. Nevertheless, they play an important role in checking the rules produced from data.

Prior knowledge can not only be used in checking the consistency of the fuzzy system, but also be incorporated in the initialization of the evolutionary algorithm. For example, some individuals can be initialized with the parameters of a standard fuzzy rule system, while the others are initialized with randomly generated numbers.

In the beginning, we generate the fuzzy rule control system using 316 groups of data collected in driving situation 1. In

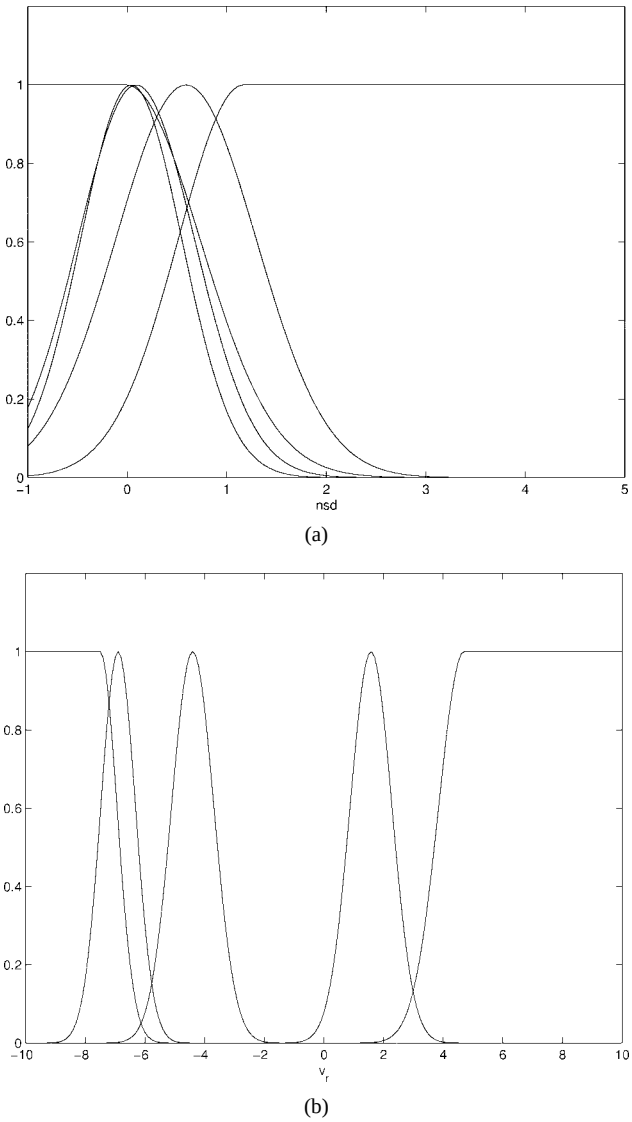


Fig. 6. Membership functions (without checking, situation 1). (a) nsd and (b) v_r.

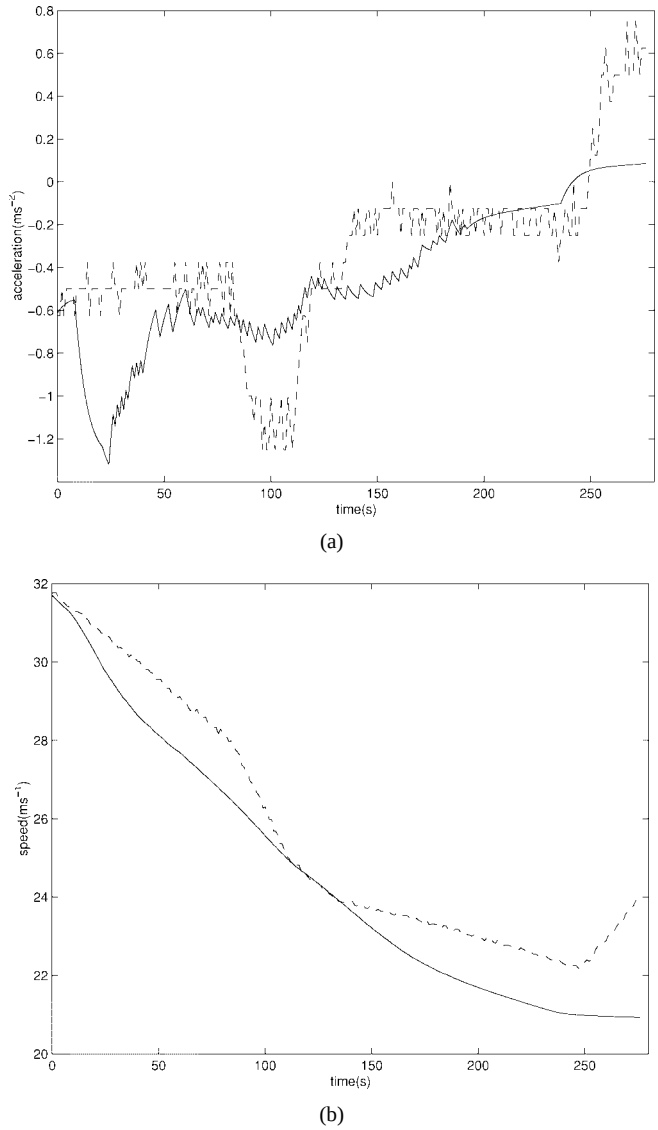


Fig. 7. Test results (without checking, situation 2). (a) Acceleration and (b) speed.

order to make comparisons, fuzzy rule systems are generated with and without completeness and consistency checking. A fuzzy rule base is first generated without checking its completeness and consistency.[2] The rule base and its inconsistency indices are provided in Tables I and II, respectively, and the speed and acceleration tracking results are illustrated in Fig. 5. In Fig. 5, the dotted lines denote the acceleration and speed measured in the experiment, and the solid lines describe the results produced by the fuzzy controller. In Fig. 5(a), the curve of the measured acceleration looks unsteady due to the discretization of the measurement. Therefore, the fuzzy controlleris not aimed to approximate the measured acceleration to every detail. This is true for the measured acceleration in the whole simulation. Taking this fact into account, we think the fuzzy controller behaves very well for the training data. The mean errors of acceleration and speed are 0.105 ms^{-2} and 0.085 ms^{-1}, respectively. That is to say, the fuzzy controller

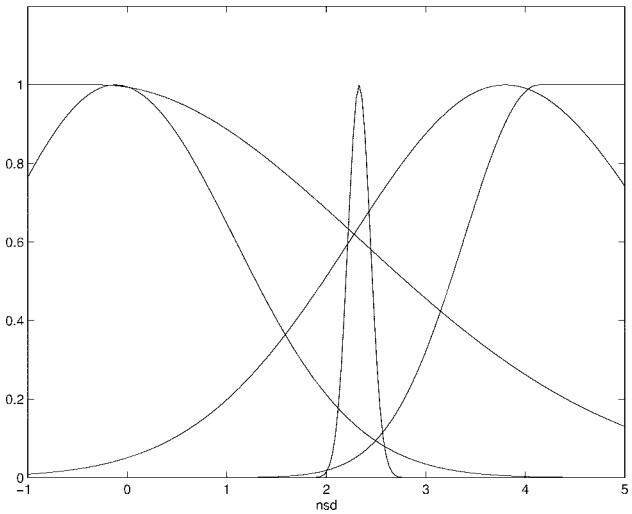
[2]Nevertheless, rules with the same IF part but different THEN parts are also avoided to assure the fairness of the comparisons.

TABLE III
RULE BASE (WITH CHECKING, SITUATION 1)

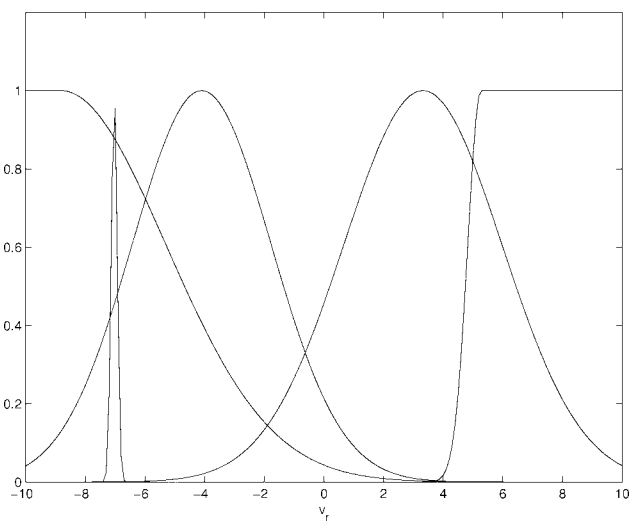
	0	1	2	3	4	5
0	*	3	4	*	*	*
1	4	*	*	*	4	7
2	*	2	2	3	*	3
3	5	7	6	*	3	3
4	*	6	*	6	*	*
5	*	*	*	*	4	3

TABLE IV
INCONSISTENCY INDEXES (WITH CHECKING, SITUATION 1)

	0	1	2	3	4	5
0	*	0.001	0.001	*	*	*
1	0.001	*	*	*	0.086	0.129
2	*	0.046	0.010	0.064	*	0.019
3	0.001	0.010	0.005	*	0.024	0.029
4	*	0.067	*	0.080	*	*
5	*	*	*	*	0.033	0.024

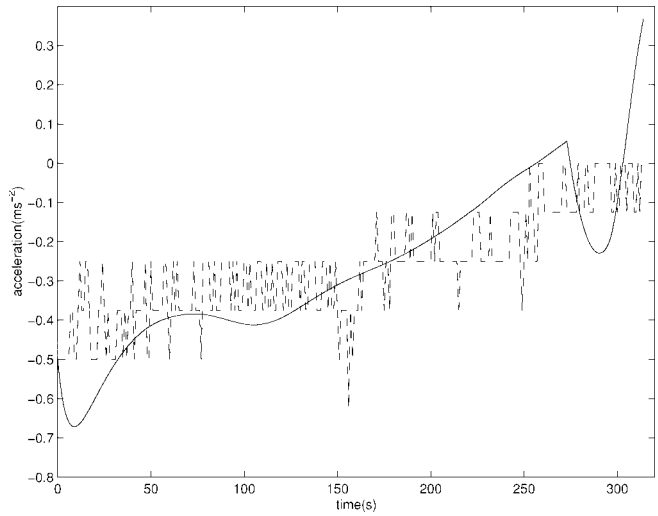


(a)

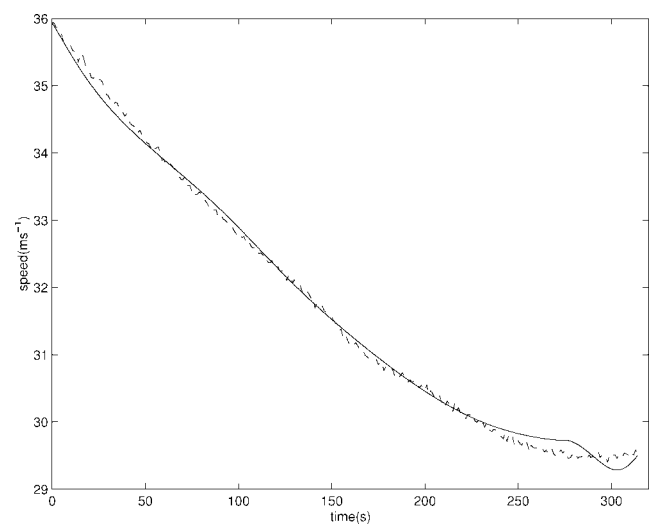


(b)

Fig. 8. Membership functions (with checking, situation 1). (a) nsd and (b) v_r.



(a)



(b)

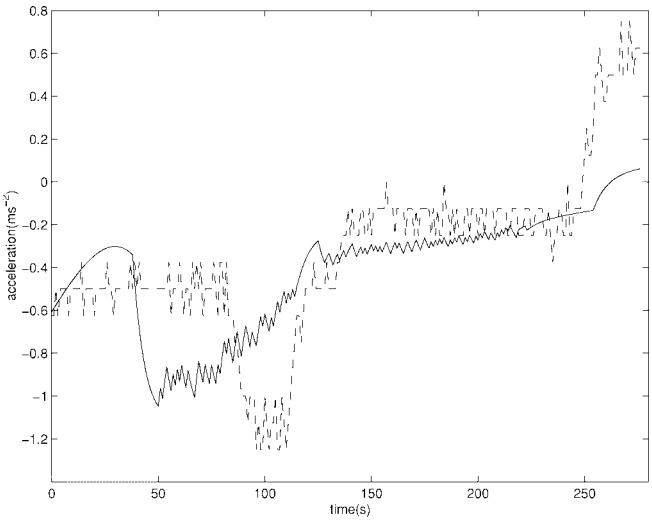
Fig. 9. Training results (with checking, situation 1). (a) Acceleration and (b) speed.

has imitated the given driving behavior very well. This can be attributed to the fact that the rule parameters and the rule structure are optimized simultaneously and the fuzzy operators are flexible. However, a fuzzy system that exhibits smart performance for training data does not necessarily perform equally well on the test data. Before we check the fuzzy system with test data, we first have a look at the membership functions (see Fig. 6). We notice that some fuzzy sets of $nsd(t)$ lack distinguishability, while the fuzzy partitioning of $v_r(t)$ is incomplete.[3] This implies that over-fitting of the membership functions has occurred. We notice further that the total degree of inconsistency is 13.4, which seems quite large. Now we evaluate the fuzzy system with 276 groups of data obtained in driving situation 2. The results are presented in Fig. 7. The mean errors of acceleration and speed are 0.233 ms^{-2} and 1.044 ms^{-1}. We note that both errors are quite large.

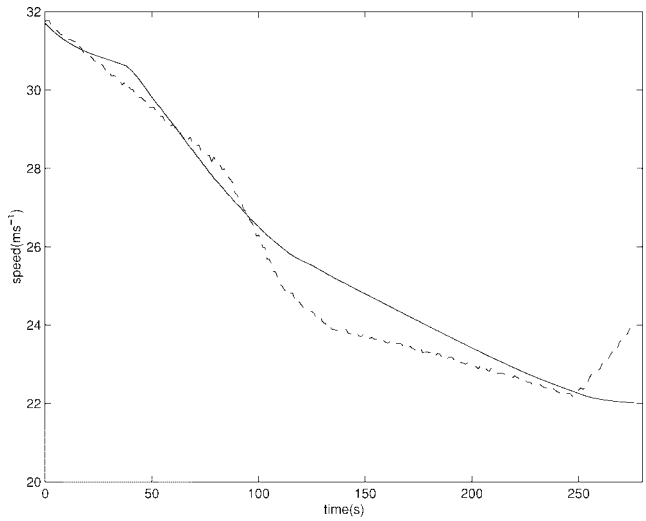
[3]Theoretically, the value of Gaussian functions never becomes zero. In practice, it will become zero due to the precision limit.

The rule base generated with completeness and consistency checking is listed in Table III, and the inconsistency indices are provided in Table IV. Note first that the fuzzy partitioning of the two input variables (see Fig. 8) are now complete and the distribution of the membership functions seems to be more reasonable. The acceleration and speed tracking results for the training data are demonstrated in Fig. 9 with mean errors of 0.107 ms^{-2} and 0.090 ms^{-1}, respectively. Compared to the fuzzy rule system generated without consistency and completeness checking, the performance for the training data is quite the same. Now we will evaluate it with the test data from driving situation 2. The test results are presented in Fig. 10 with mean errors of 0.216 ms^{-2} and 0.552 ms^{-1}, respectively. It is noticed that the mean speed error has been significantly reduced.

We also notice from Table IV that the total inconsistency index of the rule base is only 0.63. This is quite hard to believe if we do not associate the rule base with the distribution of the premise and consequent membership functions.

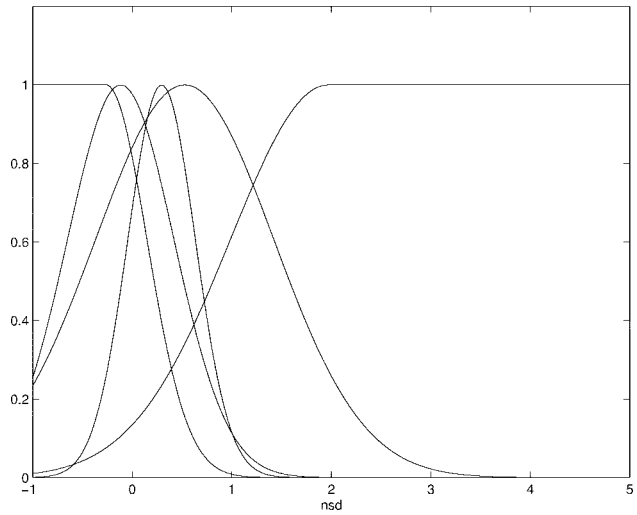


(a)

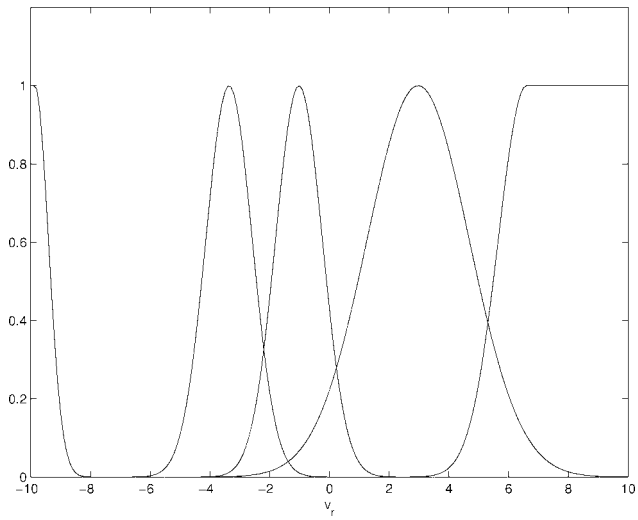


(b)

Fig. 10. Test results (with checking, situation 2). (a) Acceleration and (b) speed.



(a)

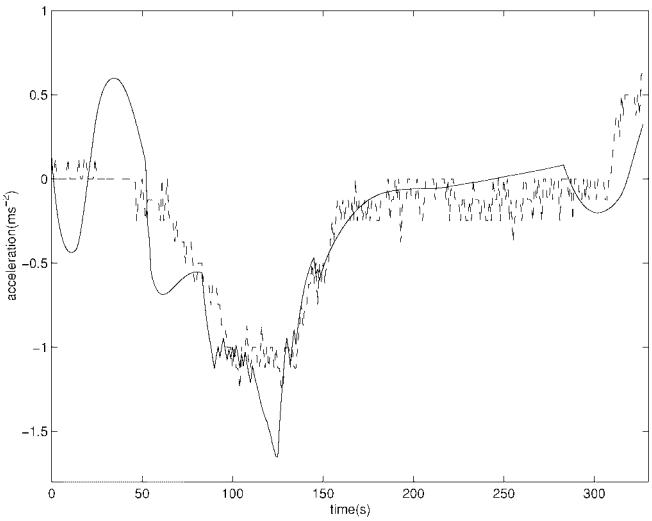


(b)

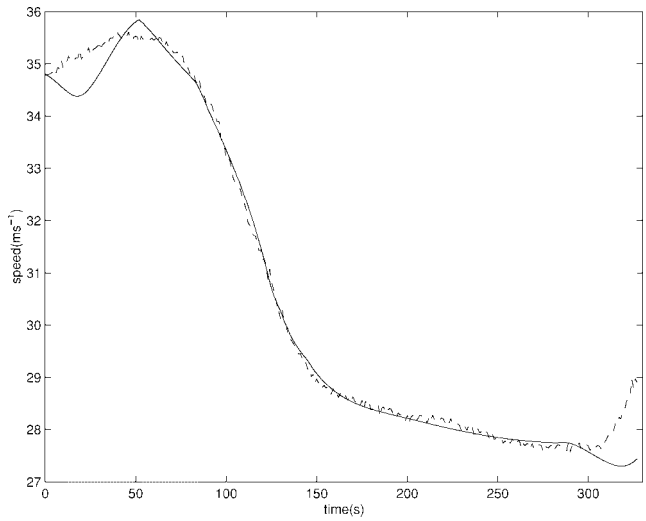
Fig. 11. Membership functions (without checking, situation 3). (a) nsd and (b) v_r.

From the above simulations, we see that a fuzzy system generated with completeness and consistency checking outperforms the fuzzy system generated without checking for test data. To confirm this conclusion, another simulation is carried out, in which the data from driving situations 3 and 4 are used as training and test data. The membership functions, training and test results of the fuzzy system generated without completeness and consistency checking are given in Figs. 11–13, respectively. The mean training errors for acceleration and speed are 0.210 ms^{-2} and 0.194 ms^{-1}, however, the mean test errors are as large as 0.348 ms^{-2} and 1.635 ms^{-1}. The fuzzy partitioning of v_r is again incomplete, and the total inconsistency index is 34.9, which denotes that the quality of the rules is not so satisfying despite the good training results. As a comparison, the membership functions, training and test results of the fuzzy system generated with completeness and consistency checking are provided in Figs. 14–16, respectively. In this case, the mean errors of acceleration and speed are 0.209 ms^{-2} and 0.843 ms^{-1} on the test samples. These values are considerably lower than those of the fuzzy system without completeness and consistency checking . The total inconsistency index reduces to 3.04.

Until now, we have shown that a FC^3 fuzzy system is superior to a conventional fuzzy system in its generalization ability if relatively few data are provided for training. In the following, we will briefly describe the simulation results when more training data are available. 1512 groups of data collected from five different driving situations are used to generate the fuzzy rule base. Similarly, we generate two fuzzy systems without and with completeness and consistency checking. The two rule bases have 17 and 15 fuzzy rules respectively. It is not surprising that no large differences are observed from training and test results of these two fuzzy systems. For the fuzzy system without completeness and consistency checking, the mean training and test errors are 0.20 ms^{-2}, 0.51 ms^{-1} and 0.26 ms^{-2}, 0.65 ms^{-1}. For the fuzzy system with completeness and consistency checking, the mean training and test errors are 0.18 ms^{-2}, 0.4 ms^{-1}

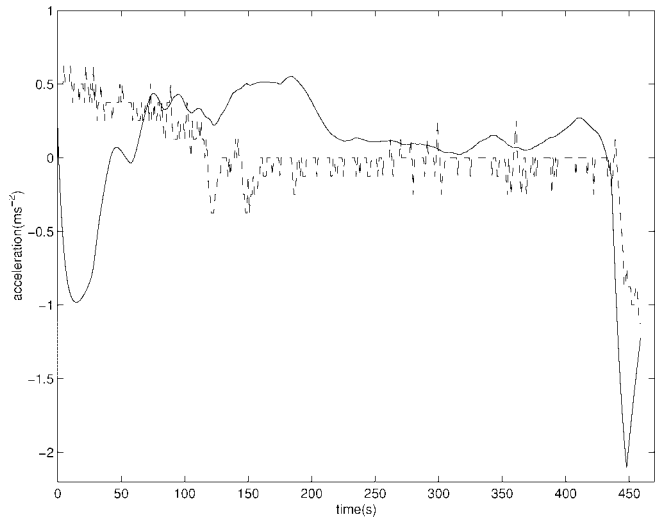


(a)

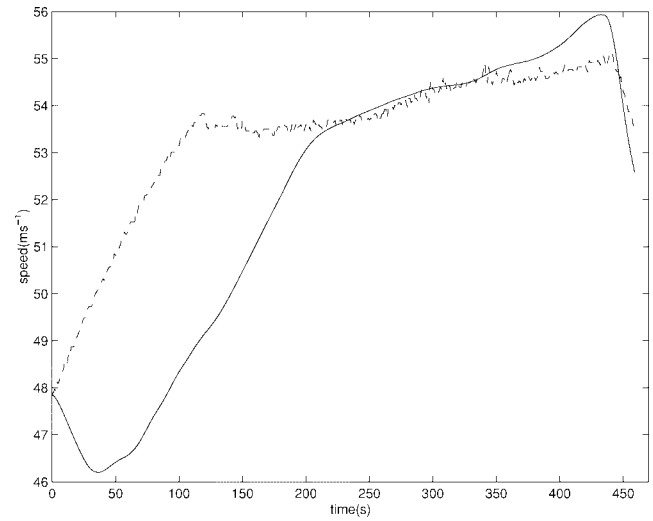


(b)

Fig. 12. Training results (without checking, situation 3). (a) Acceleration and (b) speed.



(a)



(b)

Fig. 13. Test results (without checking, Situation 4). (a) Acceleration and (b) speed.

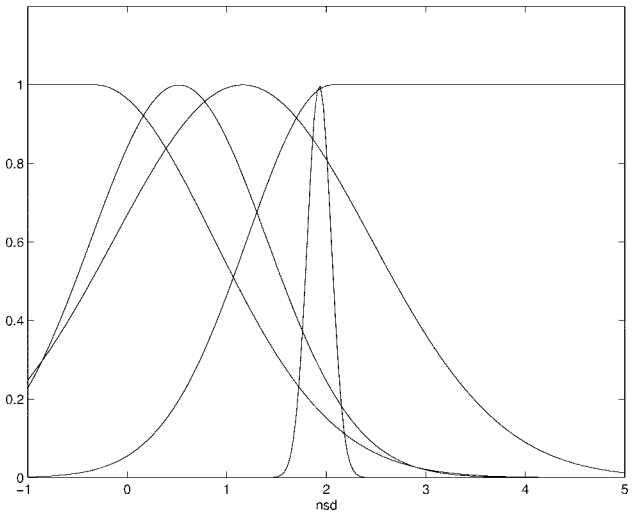
and 0.22 ms^{-2}, 0.57 ms^{-1}. This is in concordance with the fact that the generalization ability of the fuzzy systems will be better if sufficient training data are available. However, the distributions of the membership functions produced without completeness and consistency checking (Fig. 17) seem to be less promising than those of the membership functions produced with checking (Fig. 18). Moreover, the total inconsistency indexes of the checked and unchecked systems are 3.7 and 16.0, respectively. This explains possibly the reason why the test results of the unchecked system are slightly worse than the checked system. The training and test results will not be illustrated here due to space limit.

In all of these fuzzy systems, the soft T-norm and BADD defuzzification play an important role too. For example, the optimized soft coefficient α and the BADD coefficient δ in the first training example (with checking) are 0.83 and 5.23. If they are fixed to 1.0, different degrees of performance deterioration are observed in training and testing. This is true for all of the other cases.

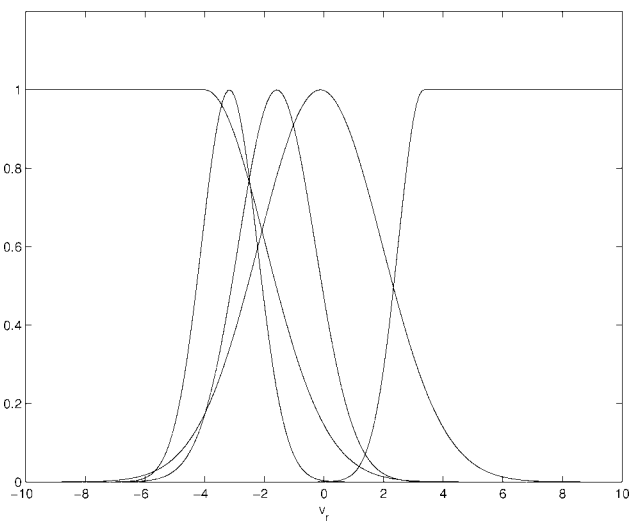
Finally, we discuss the compactness of the fuzzy systems. Although no penalty function for rule complexity is introduced in the process of rule generation, the number of fuzzy rules in all cases is reduced. This implies that a compact fuzzy system has normally better performance than a standard system. Moreover, the structures of the rule bases are all complete, although we have experienced incomplete rule structure in other cases.

V. Conclusions

A methodology for generating flexible, complete, consistent and compact fuzzy rule systems from data using evolution strategies is proposed in this paper. All the components of the fuzzy systems, including the parameters of the membership functions, the structure of the rule base and the fuzzy inference mechanisms, are encoded in a unified frame and optimized with evolution strategies. In order to evaluate the

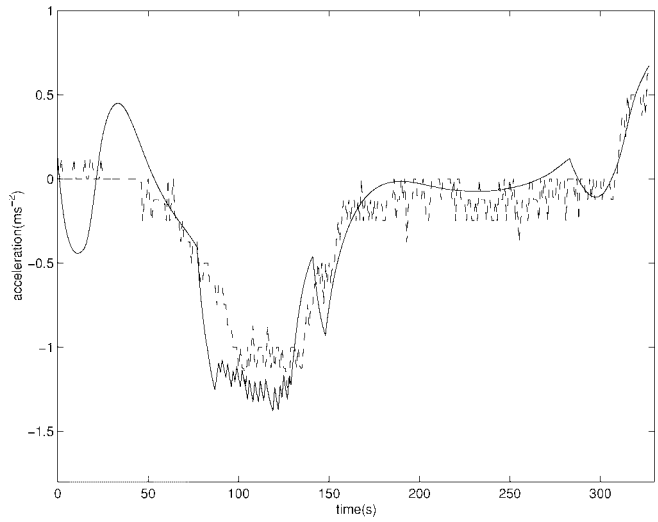


(a)

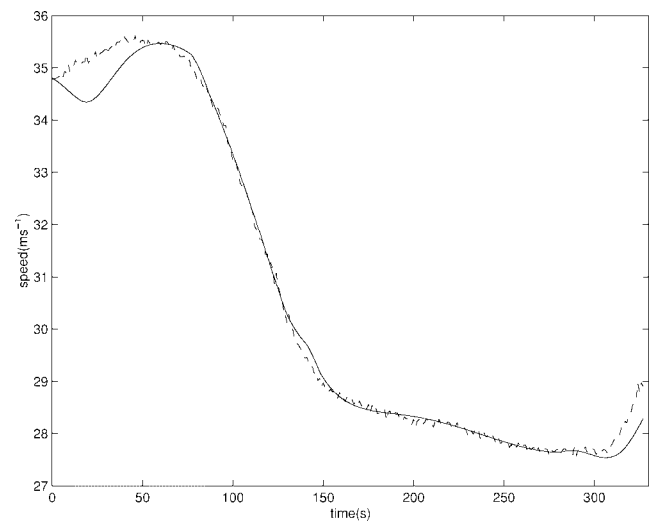


(b)

Fig. 14. Membership functions (with checking, situation 3). (a) nsd and (b) v_r.



(a)



(b)

Fig. 15. Training results (with checking, situation 3). (a) Acceleration and (b) speed.

completeness and consistency of a fuzzy system, indices for completeness and consistency are proposed with the help of the fuzzy similarity measure. These indices are integrated into the objective function so that the generated fuzzy systems are complete, and the rules are more consistent with each other and with the prior knowledge. Comparative simulation studies have been carried out to show that the fuzzy systems generated with completeness and consistency checking are advantageous over the fuzzy systems generated without completeness and consistency checking, especially when insufficient training data are available.

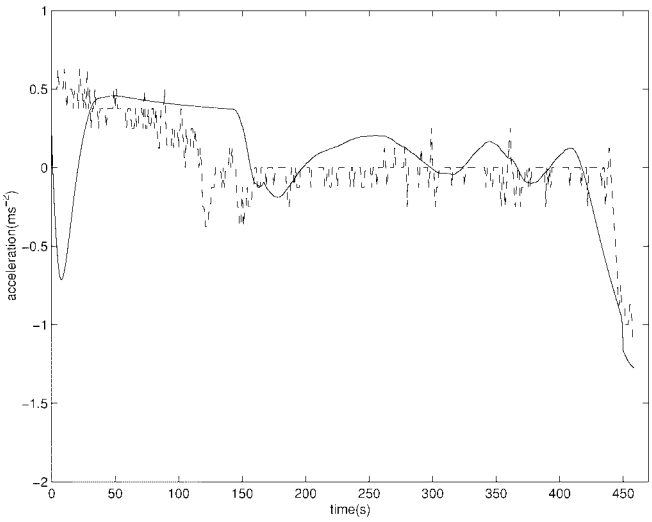
Flexibility of the fuzzy system is realized by optimizing the soft T-norm and the BADD defuzzifier. This alleviates the arduous task to select better fuzzy operators for a given problem. Since the rule structure is also evolved by the algorithm, the generated fuzzy system has always fewer rules than the standard rule system. This is imperative if the fuzzy system has more than two input variables.

APPENDIX

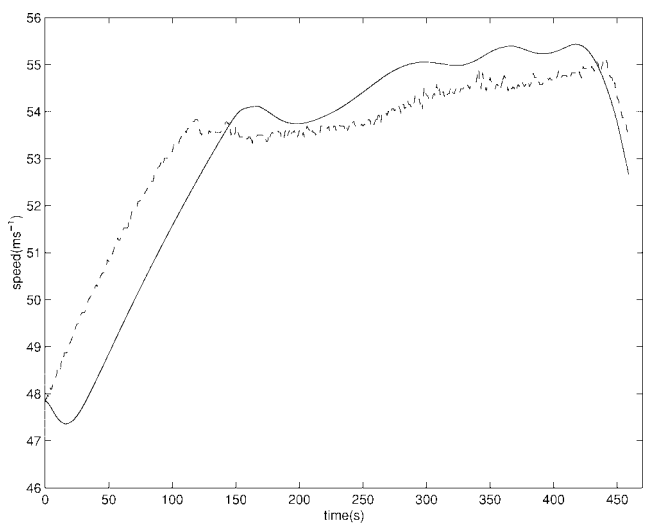
In order to compute the fuzzy similarity measure, it is necessary to calculate $M(A \cap B)$. For any two fuzzy sets described with triangular membership functions in the following form:

$$A(x)\colon\ y = \begin{cases} \dfrac{1}{m_1 - a_1}x - \dfrac{a_1}{m_1 - a_1}, & \text{if } x < m_1 \\ \dfrac{1}{m_1 - b_1}x - \dfrac{b_1}{m_1 - b_1}, & \text{if } x > m_1 \end{cases}$$
$$B(x)\colon\ y = \begin{cases} \dfrac{1}{m_2 - a_2}x - \dfrac{a_2}{m_2 - a_2}, & \text{if } x < m_2 \\ \dfrac{1}{m_2 - b_2}x - \dfrac{b_2}{m_2 - b_2}, & \text{if } x > m_2 \end{cases} \tag{31}$$

and if we suppose $m_1 < m_2$ and $b_1 > a_2$, i.e., the situation of no overlap will not be considered, then there are nine total

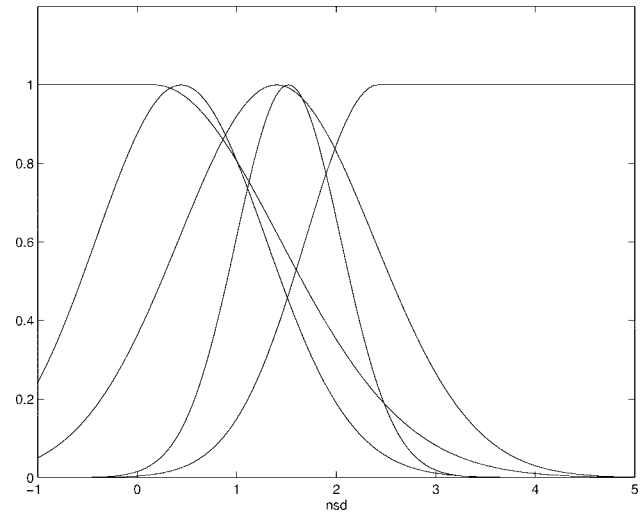


(a)

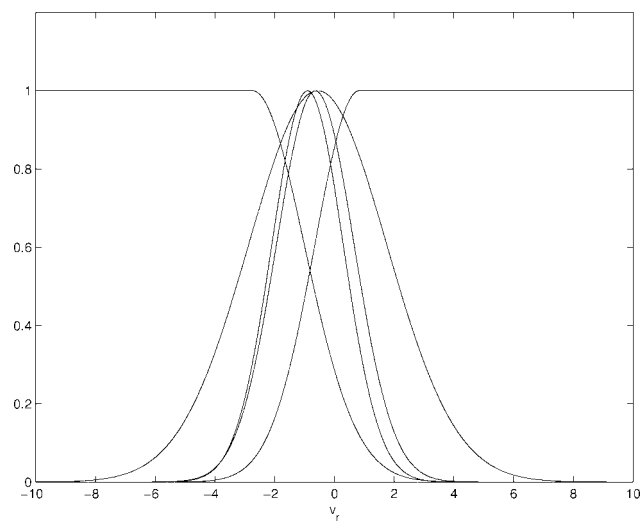


(b)

Fig. 16. Test results (with checking, situation 4). (a) Acceleration and (b) speed.



(a)



(b)

Fig. 17. Membership functions (without checking, situations 1–5). (a) nsd and (b) v_r.

possible overlapping cases in total, namely:

1) $a_1 = a_2, b_1 = b_2$;
2) $a_1 = a_2, b_1 > b_2$;
3) $a_1 = a_2, b_1 < b_2$;
4) $a_1 < a_2, b_1 = b_2$;
5) $a_1 < a_2, b_1 < b_2$;
6) $a_1 < a_2, b_1 > b_2$;
7) $a_1 > a_2, b_1 = b_2$;
8) $a_1 > a_2, b_1 > b_2$;
9) $a_1 > a_2, b_1 < b_2$.

It is noticed that these nine cases can again be classified into four situations:

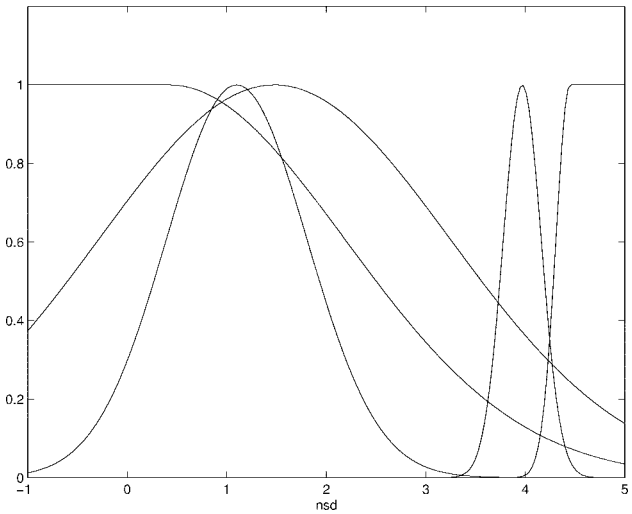
1) One intersection (noted $P_{21}(x_{21}, y_{21})$) between the right side of $A(x)$ and left side of $B(x)$, when $a_1 \leq a_2, b_1 \leq b_2$. This is true in the above case 1, 3, 4, and 5.
2) Two intersections (noted as $P_{21}(x_{21}, y_{21}), P_{22}(x_{22}, y_{22})$) between the right side of $A(x)$ and the both sides of $B(x)$, when $a_1 \leq a_2, b_1 > b_2$. This holds for the above cases 2 and 6.
3) Two intersections (noted as $P_{11}(x_{11}, y_{11}), P_{21}(x_{21}, y_{21})$) between the both sides of $A(x)$ and the left side of $B(x)$, if $a_1 > a_2, b_1 \leq b_2$. This is true for the above cases 7 and 9.
4) Three intersections (noted as $P_{11}(x_{11}, y_{11}), P_{21}(x_{21}, y_{21})$, and $P_{22}(x_{22}, y_{22})$) between the both sides of $A(x)$ and $B(x)$, if $a_1 > a_2$ and $b_1 > b_2$. See the above case 8.

In this way, $M(A \cap B)$ can be calculated according to these four cases.

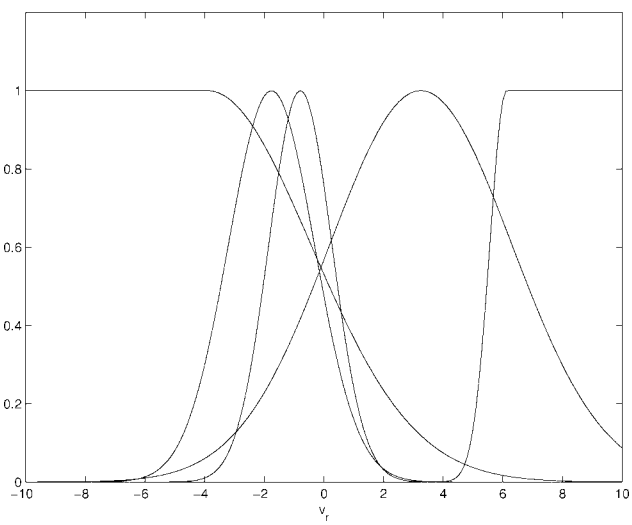
1) Case 1 (Fig. 19):

$$x_{21} = \frac{b_1 m_2 - a_2 m_1}{(b_1 - a_2) + (m_2 - m_1)},$$

$$y_{21} = \frac{b_1 - a_1}{(b_1 - a_1) + (m_2 - m_1)}.$$



(a)



(b)

Fig. 18. Membership functions (with checking, situations 1–5). (a) nsd and (b) v_r.

Thus,

$$M(A\cap B)=\int_{a_2}^{x_{21}}\left(\frac{1}{m_2-a_2}x-\frac{a_2}{m_2-a_2}\right)dx+\int_{x_{21}}^{b_1}\left(\frac{1}{m_1-b_1}x-\frac{b_1}{m_1-b_1}\right)dx=\frac{1}{2}\frac{(b_1-a_1)^2}{(b_1-a_2)+(m_2-m_1)}. \tag{32}$$

2) Case 2 (Fig. 20):

x_{21} and y_{21} are the same as in case 1, and

$$x_{22}=\frac{b_1m_2-m_1b_2}{(b_1+m_2)+(m_2-m_1)},$$
$$y_{22}=\frac{b_1-b_2}{(b_1-b_2)+(m_2-m_1)}.$$

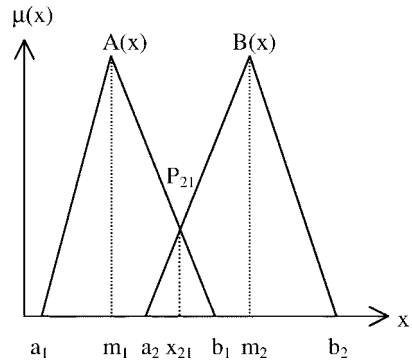


Fig. 19. Case 1.

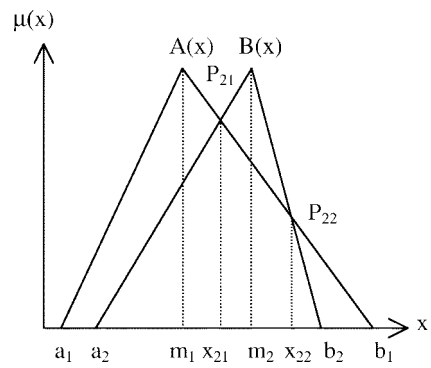


Fig. 20. Case 2.

Similarly,

$$M(A\cap B)=\int_{a_2}^{x_{21}}\left(\frac{1}{m_2-a_2}x-\frac{a_2}{m_2-a_2}\right)dx+\int_{x21}^{x_{22}}\left(\frac{1}{m_1-b_1}x-\frac{b_1}{m_1-b_1}\right)dx+\int_{x_{22}}^{b_2}\left(\frac{1}{m_2-b_2}x-\frac{b_2}{m_2-b_2}\right)dx=\frac{1}{2}\frac{(b_1-a_2)^2}{(b_1-a_2)+(m_2-m_1)}-\frac{1}{2}\frac{(b_1-b_2)^2}{(b_1-b_2)+(m_2-m_1)}. \tag{33}$$

3) Case 3 (Fig. 21):

$$x_{11}=\frac{a_1m_2-a_2m_1}{(a_1-a_2)+(m_2-m_1)}$$
$$y_{11}=\frac{a_1-a_2}{(a_1-a_2)+(m_2-m_1)}.$$

x_{21} and y_{21} are the same as in case 1. Then we have,

$$M(A\cap B)=\int_{a_1}^{x_{11}}\left(\frac{1}{m_1-a_1}x-\frac{a_1}{m_1-a_1}\right)dx+\int_{x_{11}}^{x_{21}}\left(\frac{1}{m_2-a_2}x-\frac{a_2}{m_2-a_2}\right)dx+\int_{x_{21}}^{b_1}\left(\frac{1}{m_1-b_1}x-\frac{b_1}{m_1-b_1}\right)dx=\frac{1}{2}\frac{(b_1-a_2)^2}{(b_1-a_2)+(m_2-m_1)}-\frac{1}{2}\frac{(a_1-a_2)^2}{(a_1-a_2)+(m_2-m_1)}. \tag{34}$$

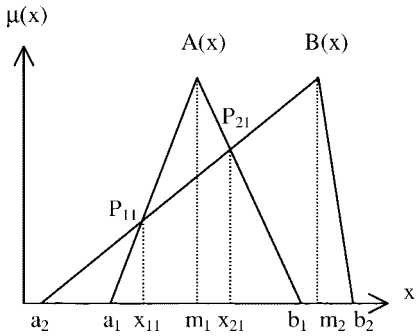


Fig. 21. Case 3.

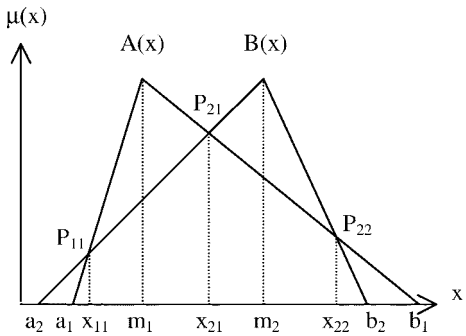


Fig. 22. Case 4.

4) Case 4 (Fig. 22).

All the coordinator values of the intersection points are now available, and consequently, $M(A \cap B)$ is calculated by

$$\begin{aligned} M(A\cap B) &= \int_{a_1}^{x_{11}} \left(\frac{1}{m_1-a_1}x - \frac{a_1}{m_1-a_1}\right) dx \\ &\quad + \int_{x_{11}}^{x_{21}} \left(\frac{1}{m_2-a_2} - \frac{a_2}{m_2-a_2}\right) dx \\ &\quad + \int_{x_{21}}^{x_{22}} \left(\frac{1}{m_1-b_1}x - \frac{b_1}{m_1-b_1}\right) dx \\ &\quad + \int_{x_{22}}^{b_2} \left(\frac{1}{m_2-b_2}x - \frac{b_2}{m_2-b_2}\right) dx \\ &= \frac{1}{2}\frac{(b_1-a_2)^2}{(b_1-a_2)+(m_2-m_1)} \\ &\quad - \frac{1}{2}\frac{(a_1-a_2)^2}{(a_1-a_2)+(m_2-m_1)} \\ &\quad - \frac{1}{2}\frac{(b_1-b_2)^2}{(b_1-b_2)+(m_2-m_1)}. \end{aligned} \tag{35}$$

If the membership functions are Gaussian functions, they can be approximated by isosceles triangular membership functions [16] for the sake of computational simplicity. Given two Gaussian membership functions (c_1, w_1) and (c_2, w_2), where c_1, c_2, and w_1, w_2 are the centers and widths respectively, then they can be replaced by two triangulars with $a_i = c_i - w_i\sqrt{\pi}, b_i = c_i + w_i\sqrt{\pi}, i = 1, 2$. It is straightforward that they can be calculated using one of the equations provided in case 1, 2, or 3. It is noticed that, case 4 does not happen for the isosceles triangulars.

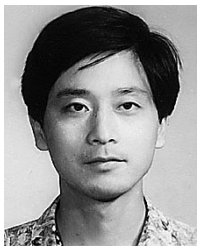
Acknowledgment

The authors would like to thank Dr. W. Wienholt and Dr. C. Goerick for their kind help.

References

[1] T. Bäck, "Parallel optimization of evolutionary algorithms," in *Parallel Problem Solving from Nature*. Berlin, Germany: Springer-Verlag, 1994, pp. 418–427.
[2] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evol. Computat.*, vol. 1, no. 1, pp. 1–23, 1993.
[3] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy controllers through reinforcement," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 724-739, 1992.
[4] J. J. Buckley, "Universal fuzzy controllers," *Automatica*, vol. 28, no. 6, pp. 1245–1248, 1992.
[5] M. G. Cooper and J. J. Vidal, "Genetic design of fuzzy controllers," in *Proc. 2nd Int. Conf. Fuzzy Theory Technology*, Durham, NC, 1993.
[6] C. Elkan, "The paradoxical success of fuzzy logic," in *Proc. 11th Nat. Conf. Artificial Intelligence*, 1993, pp. 698–703.
[7] M. M. Gupta and J. Qi, "Design of fuzzy logic controllers based on generalized T-operators," *Fuzzy Sets Syst.*, vol. 40, pp. 473–489, 1991.
[8] K. J. Hunt, R. Haas, and M Brown, "On the functional equivalence of fuzzy inference systems and spline-based networks," *Int. J. Neural Syst.*, vol. 6, no. 2, pp. 171–184, 1995.
[9] J.-S. R. Jang and C. T. Sun, "Self-learning fuzzy controller based on temporal back-propagation," *IEEE Trans. Neural Networks*, vol. 3, pp. 714–723, 1992.
[10] Y. Jin, "Decentralized adaptive fuzzy control of robot manipulators," *IEEE Trans. Syst., Man, Cybern. B*, vol. 28, pp. 47–58, Feb. 1998.
[11] Y. Jin, J. Jiang, and J. Zhu, "Neural network based fuzzy identification with application to modeling and control of complex systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 990–997, June 1995.
[12] Y. Jin and W. v. Seelen, " Evaluating flexible fuzzy controllers via evolution strategies," *Fuzzy Sets Syst.*, vol. 108, pp. 243–252, 1999.
[13] C. Karr, "Applying genetics to fuzzy logic," *IEEE AI Expert*, vol. 6, no. 2, pp. 26–33, 1992.
[14] M. A. Lee and H. Takagi, "Integrating design stages of fuzzy systems using genetic algorithms," in *Proc. 2nd IEEE Conf. Fuzzy Systems*, 1993, vol. 1, pp. 612–617.
[15] D. D. Leitch, "A new genetic algorithm for the evolution of fuzzy systems," Ph.D. dissertation, Dept. Eng. Sci., Univ. Oxford, Oxford, U.K., 1995.
[16] C. T. Lin and C. S. G. Lee, "Real-time supervised structure/parameter learning for fuzzy neural network," in *Proc. IEEE Conf. Fuzzy Systems*, 1992, pp. 1283–1290.
[17] ——, "Neural-network-based fuzzy logic control systems," *IEEE Trans. Comput.*, vol. 40, pp. 1320–1336, Dec. 1991.
[18] D. A. Linkens and H. O. Hyongesa, "Genetic algorithm for fuzzy control. Part 1: Off-line system development and application," *Proc. Inst. Elect. Eng. Control Theory Applications*, vol. 142, pp. 171–176, 1995.
[19] R. Marks, "Intelligence: Computational versus artificial," *IEEE Trans. Neural Networks*, vol. 4, no. 5, pp. 737–739, 1993.
[20] J. Nie and D. A. Linkens, "Fast self-learning multivariable fuzzy controllers constructed from a modified CPN network," *Int. J. Contr.*, vol. 60, no. 3, pp. 369–393, 1994.
[21] D. Park, A. Kandel, and G. Langholz, "Genetic-based new fuzzy reasoning models with application to fuzzy control," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 39–47, Jan. 1994.
[22] T. A. Runkler, "Selection of appropriate defuzzification methods using application specific properties," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 1, pp. 72–79, 1997.
[23] H. P. Schwefel, *Evolution and Optimum Seeking*. New York: Wiley, 1995.
[24] M. Sugeno and K. Tanaka, "Successive identification of a fuzzy model and its application to prediction of complex systems," *Fuzzy Sets Syst.*, vol. 42, pp. 315–324, 1994.
[25] L. X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*. Englewood Cliffs, NJ: Prentice-Hall,1994.
[26] L. X. Wang and M. Mendel "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1414–1427, 1992.
[27] W. Wienholt, "Optimizing the structure of RBF networks by optimizing fuzzy inference systems with evolution strategy," Internal Rep., Inst. Neuroinformatik, Ruhr-Universität Bochum, 1993.

[28] R. R. Yager and D. P. Feliv, "Analysis of flexible structured fuzzy logic controller," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 1035–1043, 1994.
[29] R. R. Yager and L. A. Zadeh, Eds., *Fuzzy Sets, Neural Networks, and Soft Computing*. New York: Van Nostrand Reinhold, 1994.
[30] T. Yamaguchi, T. Takagi, and T. Mita, "Self-organizing control using fuzzy neural networks," *Int. J. Contr.*, vol. 56, no. 2, pp. 415–439, 1992.



Yaochu Jin (M'98) was born in Wujiang, China, in 1966. He received the degrees in electrical engineering from Zhejiang University, Hangzhou, China.

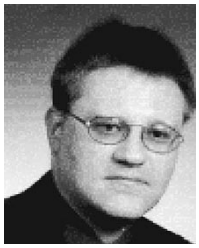
He joined the Electrical Engineering Department, Zhejiang University, in 1991, where he became an Associate Professor in 1996. He was a Visiting Researcher and then a Scientific Staff Member (Wissenschaftlicher Mitarbeiter) at the Institut für Neuroinformatik, Ruhr-Universität Bochum, Germany, from 1996 to 1998. Currently, he is a Postdoctoral Associate in the Department of Industrial Engineering, Rutgers University, New Brunswick, NJ, and Research and Technology, Allied Signal, Inc., Morristown, NJ. He has published more than 20 technical papers and co-authored two books. His current research interest is mainly on the control and optimization of complex systems using fuzzy set theory, neural networks, and evolutionary computations.

Mr. Jin received the Science and Technology Progress Award from the State Education Commission (now Ministry of Education) of China in 1995.

Werner von Seelen received the Dipl.Ing. degree and the Doctorate degree from the Faculty of Electrical Engineering, Technische Universität Hannover, Hannover, Germany.

He joined the Fraunhofer Institute and headed a group for biological information processing. After a two-year period with the Batelle Institute, working in nonlinear optics, he became a Professor with the Department of Biophysics, Universität Mainz, Mainz, Germany. Since 1989, he has been Chairman for Theoretical Biology at Ruhr-Universität Bochum. He is Professor and Head of the Department of Theoretical Biology with the Institute of Neuroinformatik, Ruhr-Universität Bochum, Bochum, Germany, and Director of the Centre for Neuroinformatik. His current research focuses on system theory, brain research, and a neural architecture for autonomous systems. He has written numerous research articles and was the editor/co-editor of several books. He is a member of the editorial board of the journal *Neural Networks*.

Dr. von Seelen was President of the German Society of Cybernetics for three years and is a member of several scientific societies. In 1995, he was awarded the Kupfmüller Ring and, in 1998, the Karl-Heinz Beckurts Prize for his outstanding scientific contributions in the field of interdisciplinary research.



Bernhard Sendhoff studied physics at the Ruhr-Universität Bochum, Bochum, Germany, and the University of Sussex, Sussex, U.K. In 1993, he received the Dipl. degree and, in 1998, the Doctorate degree in physics from the Faculty of Physics and Astronomy, Ruhr-Universität Bochum.

Since 1994, he has been a Researcher at the Institute of Neuronformatik. He is author/co-author of several scientific papers and co-editor of the 1996 *ICANN Proceedings*, of which he was a member of the organization team. He is the Head of the Optimization Group, Department of Theoretical Biology, Institute of Neuroinformatik, Ruhr-Universität Bochum. His current research interests include the design and structure optimization of adaptive systems based on neural networks, fuzzy systems and evolutionary algorithms.

Dr. Sendhoff is a member of the ENNS.