ELSEVIER

Fuzzy Sets and Systems 132 (2002) 217–231

FUZZY sets and systems

www.elsevier.com/locate/fss

Reduction of fuzzy control rules by means of premise learning – method and case study

N. Xiong *, L. Litz

Institute of Process Automation, Department of Electrical and Computer Engineering, University of Kaiserslautern, Germany

Received 12 March 1999; received in revised form 11 September 2001; accepted 19 January 2002

Abstract

Rule number reduction is important for fuzzy control of complex processes with high dimensionality. It is stated in the paper that this issue can be treated effectively by means of learning premises with general structure. Since conditions of rules are generalised by a genetic algorithm (GA) rather than enumerated according to every AND-connection of input fuzzy sets, a parsimonious knowledge base with a reduced number of rules can be expected. On the other hand, to give a numerical evaluation of possible conflicts among rules, a consistency index of the rule set is established. This index is integrated into the fitness function of the GA to search for a set of optimal rule premises yielding not only good control performance but also little or no inconsistency in the fuzzy knowledge base. The advantage of the proposed method is demonstrated by the case study of development of a compact fuzzy controller to balance an inverted pendulum in the laboratory.

Keywords: Fuzzy controller; Rule premise; Membership function; Genetic algorithm; Premise learning

1. Introduction

Fuzzy controllers are knowledge-based systems which provide a framework for formalising intuition and experiences of human experts and operators [17]. Based on Zadeh's theory of fuzzy sets [28], a typical fuzzy controller [6] maintains a rule base of fuzzy rules and associated fuzzy sets for mapping real-numbered inputs to outputs. Definition of a control rule usually involves specification of the rule antecedent and determination of the corresponding consequence which is suggested as controller output under the specified premise. Rule premises are significant since they correspond to partitions of the input domain and determine the structure of a knowledge base.

A common practice in traditional approaches to building rule bases is to first use canonical AND-connections of input fuzzy sets as rule conditions and then to determine their consequent counterparts. In this manner the size (rule number) of a generated rule base is given by the expression $\prod_{i=1}^{n} q[i]$, where n is the number of controller inputs and $q[i]$ is the number of linguistic values defined for the ith input. This presents no problems in cases of n being small, but would result in a large quantity of rules when

* Corresponding author. Center for Autonomous Systems, Royal Institute of Technology, 10044 Stockholm, Sweden.
E-mail address: n.xiong@ieee.org (N. Xiong).

0165-0114/02/$ - see front matter
PII: S0165-0114(02)00112-4

dealing with high-dimensional control problems. An extremely high number of rules could undermine real-time control performance in addition to causing huge memory requirement in the implementation. Furthermore, a knowledge base with too many detailed rules is not favourable for human users to check and understand the content of it.

We believe that, for complex control problems with many input variables, it is not realistic to account for all AND-combinations of input fuzzy sets in constructing a rule base, as the number of such combinations increases exponentially with the input number. To reduce the size of the rule base, more general premise structure than canonical AND-connections is thus needed. On the other side, introducing general premise structure may lead to the problem of knowledge conflict. It is difficult to validate the correct interactions among these general rules to maintain a linguistic consistency of the knowledge base [16].

This paper aims to establish general premises of rules using a genetic algorithm (GA) for arriving at a compact knowledge base. The point of departure is the consideration that a control rule essentially gives the description of a situation under which a special fuzzy action should be performed. With this viewpoint, we suggest searching in the input domain for appropriate conditions for different conclusions to build a rule base. For instance, it can be explored which input situations correspond to the "middle" output of the controller. It is not necessary to enumerate all combinations of input linguistic terms in the knowledge base. Other premise structure is often preferred to enable bigger coverage of the input domain by an individual rule. Fig. 1 shows the overall structure of our scheme for designing a fuzzy controller by means of premise learning. The upper limits of the rule numbers for different conclusions are predetermined by human users according to the problem at hand. These upper limits can be considered as an estimation of the sufficient amounts of rules to satisfy the control goal. GA is utilised here as a learning mechanism to search for optimal premises of such rules with fixed conclusions and to optimise fuzzy set membership functions at the same time. It must be noted that, as a result, the learned rule base will not necessarily have exactly the same number of rules for an output fuzzy set as the upper limit presumed by users. As will be explained in Section 4, owing to possible invalid premises coded in genetic strings, the actual rule set learned is likely to be smaller than what is assumed beforehand. An adaptation of the rule numbers for different fuzzy conclusions is thus enabled in the sense that we do not have to prescribe them accurately in advance.

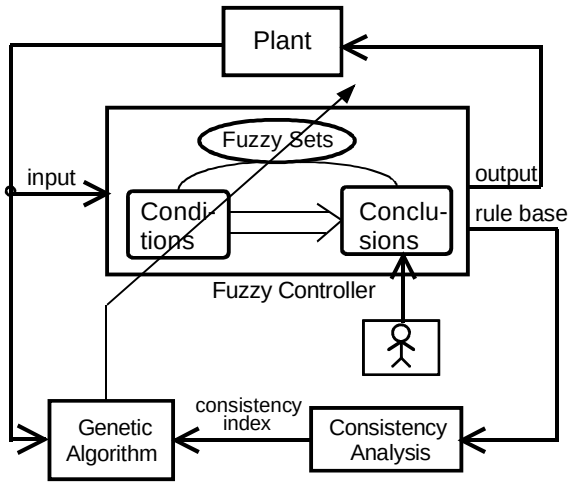

Fig. 1. Overall structure of the design scheme for a fuzzy controller.

On the other side, there will be contradictory among rules with different consequences if their conditions have linguistic overlapping. It is critical for the learning mechanism to identify and evaluate possible conflicts in a rule base to get to a solution with good logical coherence. For this purpose, a numerical assessment named as "consistency index" of the rule base is established. This index is integrated into the fitness function of the GA to search for a set of optimal rule premises yielding not only good control performance but also little or no inconsistency in the fuzzy knowledge base.

As a case study, the proposed method was applied to the problem of balancing an inverted pendulum in the laboratory. In addition to its very satisfactory control capability, our designed fuzzy controller has a much smaller rule base compared with other reported fuzzy controllers on similar (inverted pendulum) plants. This demonstrates the merit of our method to reduce the number of rules for developing compact fuzzy control systems.

2. Previous works of fuzzy controller design using genetic algorithms

Applications of GAs to facilitate automatic generation of optimal fuzzy controllers have been studied by many researchers since early 1990s. These previous investigations can be summarised into the following five categories:

(1) Optimising fuzzy set membership functions under a set of fully defined fuzzy if–then rules, see the works by Karr [12,13,14].
(2) Learning rule conclusions only, with fixed antecedents and membership functions set by hand, see [4,21,25] as examples.
(3) Learning both rule conclusions and fuzzy set membership functions but in stages, with a fixed set of rule antecedents determined by hand. This means first evolving good fuzzy rule conclusions using fixed membership functions and premises, and then fine tuning membership functions under fixed good fuzzy rule sets. Examples of the works in this class can be found in [15,26].
(4) Learning both fuzzy rule conclusions and fuzzy set membership functions simultaneously, while rule conditions are specified by users in advance, see examples from [7,10,18,19,22,24,27].
(5) Simultaneous evolving membership functions and their relations, see [3,11].

A common feature of the works in the first four categories is that the premise structure is fixed in advance and only membership functions and/or consequences of rules are optimised by GA. In fact, they all take into account all canonical AND-connections of input fuzzy sets as rule conditions. As the number of rules in such cases increases exponentially with the number of input variables, these methods are not expected to scale well to high-dimensional problems.

The approaches falling into the fifth category aim at detecting suitable membership functions on both sides of a rule. Rules are represented in terms of centres and widths of fuzzy set membership functions they relate. That means, each rule R_k for an n-input, m-output system is expressed as

$$R_k : (x_{c1k}, x_{w1k}); \cdots; (x_{cnk}, x_{wnk}) \Rightarrow (y_{c1k}, y_{w1k}); \cdots; (y_{cmk}, y_{wmk}).$$

The bracketed terms denote the centres and widths of fuzzy set membership functions over the range of input and output variables. The condition part of each rule comprises one membership function for every input variable and the conclusion part comprises one membership function for each output variable. The genome representing the whole rule base is a concatenated string of such fuzzy rules, allowing genetic operators to work on both fuzzy rule set and membership functions at the same time. This representation offers powerful expressive power for the learning of rules as well as the benefit for creating a compact rule base when scaling to multi-dimensional problems. Unfortunately, such advantages are achieved at the cost of using localised fuzzy sets, i.e., the membership functions pertaining to individual rules rather than global fuzzy sets used by all rules. Clearly, rules no longer employ easily understood fuzzy sets such as “Big” or “Small”, so that the rule set developed seems more like a black box and linguistic interpretation of the rules is difficult, if not impossible.

This paper contributes a new alternative for the synthesis of GA and fuzzy control systems. The proposed method distinguishes itself from previous works in its ability to optimise rule premises with general structure such that a compact and comprehensible rule base can be expected as the result of the genetic learning.

3. Multiple-term unified fuzzy rules and consistency evaluation

Suppose a fuzzy control system with $X = (x_1, x_2, \ldots, x_n)$ as its inputs and y as its output. Each input x_j ($j = 1, \ldots, n$) has $q[j]$ linguistic terms denoted as $A(j,1)$, $A(j,2), \ldots, A(j,q[j])$. V_j is defined as a vector, the elements of which are the fuzzy sets corresponding to input x_j, thus we can write: $V_j = (A(j,1), A(j,2), \ldots, A(j,q[j]))$. A universal rule in a knowledge base has the form as

$$\text{if } [x_{p(1)} = F_1(V_{p(1)})] \text{ and } [x_{p(2)} = F_2(V_{p(2)})] \text{ and} \ldots \text{and } [x_{p(s)} = F_s(V_{p(s)})] \quad \text{then } y = B. \quad (1)$$

Here B is a reference output fuzzy set for this rule. $p(\cdot)$ is an integer function mapping from $\{1, 2, \ldots, s\ (s \leqslant n)\}$ to $\{1, 2, \ldots, n\}$ satisfying $\forall x \neq y$, $p(x) \neq p(y)$. F_k ($k = 1, \ldots, s$) represents a logical connection among some

fuzzy sets in the vector $V_{p(k)}$. If the premise includes all input variables in it (e.g. $s=n$), we say that this rule has a complete structure, otherwise its structure is incomplete.

In principle, a rule base for the fuzzy controller can contain arbitrary rules in the form of (1). In practice, however, only some special forms of universal rules are often used. This paper is concentrated on learning conditions of the so-called "*multiple-term unified fuzzy rules*", whose definition is given below:

Definition 1. The rule expressed in (1) is called a multiple-term unified fuzzy rule if the following condition holds:

$$\forall k \in \{1,2,\ldots,s\}, \quad \exists D(k) \subset \{1,2,\ldots,q[p(k)]\}$$
$$\text{such that } F_k(V_{p(k)}) = \bigcup_{j\in D(k)} A(p(k),j). \tag{2}$$

Multiple-term unified fuzzy rules are based on multiple-value fuzzy logic. An important character of such rules is that an union operation of input fuzzy sets is allowed to appear in their premises. If the rule in (1) satisfies Definition 1, it can be rewritten as

$$\begin{aligned} \text{if } & \left[x_{p(1)} = \bigcup_{j\in D(1)} A(p(1),j)\right] \\ \text{and } & \left[x_{p(2)} = \bigcup_{j\in D(2)} A(p(2),j)\right] \quad \text{and} \cdots \\ \text{and } & \left[x_{p(s)} = \bigcup_{j\in D(s)} A(p(s),j)\right] \quad \text{then } y = B. \end{aligned} \tag{3}$$

Definition 2. The rule expressed in (1) is termed as a single-term connected fuzzy rule if the following condition holds:

$$\forall k \in \{1,2,\ldots,s\}, \quad \exists j_k \in \{1,2,\ldots,q[p(k)]\}$$
$$\text{such that } F_k(V_{p(k)}) = A(p(k),j_k). \tag{4}$$

Single-term connected rules defined above contain only one linguistic term for every included input variable in their premises. They can be regarded as a "degeneration" from the multiple-term unified rules in (3), if all the integer sets $D(i)$ $(i=1,\ldots,s)$ consist of only one element.

Definition 3. The rule expressed in (1) is an elementary fuzzy rule if the following conditions hold:

$$(1)\ s = n, \tag{5}$$

$$(2)\ \forall k \in \{1,2,\ldots,n\}, \exists j_k \in \{1,2,\ldots,q[p(k)]\}$$
$$\text{such that } F_k(V_{p(k)}) = A(p(k),j_k). \tag{6}$$

Clearly, elementary fuzzy rules are single-term connected rules which have complete structure. Using canonical AND-connections of linguistic terms as rule premises, elementary fuzzy rules are especially suitable to describe strategies to control simple plants with low input dimensions.

For fuzzy control of complex processes with high input dimensions, general multiple-term unified rules are preferable, since they achieve bigger coverage of input domain compared with elementary rules. A rule with incomplete structure or containing OR connections of linguistic terms can take the place of a set of related elementary rules, as illustrated in the following two examples:

Example 1. If $(x1=NZ$ or $PZ)$ and $(x2=NZ$ or $PZ)$ then $U=$ Small. With the premise illustrated in Fig. 2, the rule covers four elementary rules

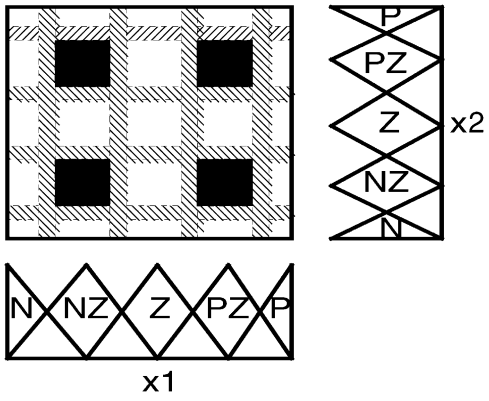

Fig. 2. Rule premise in Example 1.

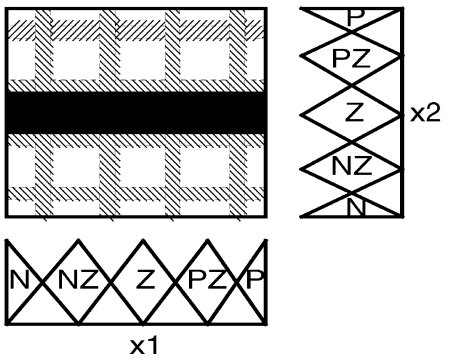

Fig. 3. Rule premise in Example 2.

listed below:

(1) If $(x1=NZ)$ and $(x2=NZ)$ Then $U=$Small.
(2) If $(x1=NZ)$ and $(x2=PZ)$ Then $U=$Small.
(3) If $(x1=PZ)$ and $(x2=NZ)$ Then $U=$Small.
(4) If $(x1=PZ)$ and $(x2=PZ)$ Then $U=$Small.

Example 2. If $(x2=Z)$ then $U=$Big. With the incompletely structured premise shown in Fig. 3, this rule covers the following group of elementary rules:

(1) If $(x1=N)$ and $(x2=Z)$ Then $U=$Big.
(2) If $(x1=NZ)$ and $(x2=Z)$ Then $U=$Big.
(3) If $(x1=Z)$ and $(x2=Z)$ Then $U=$Big.
(4) If $(x1=PZ)$ and $(x2=Z)$ Then $U=$Big.
(5) If $(x1=P)$ and $(x2=Z)$ Then $U=$Big.

However, there exists also a negative effect of using multiple-term unified fuzzy rules in the sense that consistency of the rule base is not so easily guaranteed as with using elementary rules. Linguistic overlapping between the premises of two rules could lead to a conflict between them, if the two rules suggest distinct output fuzzy sets in their conclusions. In order to give a numerical assessment of knowledge coherence of the rule base, several definitions are introduced below.

Definition 4. The input pattern (IP) of a multiple-term unified fuzzy rule in (3) is the following set of linguistic term connections:

$$IP=\{A(1,m_1)\cap A(2,m_2)\cap\cdots\cap A(n,m_n)\mid m_1\in F(1),m_2\in F(2),\ldots,m_n\in F(n)\},\quad(7)$$

where $F(i)$ are sets of integers defined as follows:

$$F(i)=\begin{cases}D(p^{-1}(i)) & \text{if } p^{-1}(i)\neq\emptyset,\\ \{1,2,\ldots,q[i]\} & \text{in other cases}\end{cases}\quad(i=1,2,\ldots,n).\quad(8)$$

Actually, the input pattern defined for a general rule is composed of premises of the elementary rules which are covered by it. For example, the input pattern of the rule in Example 1 is $\{NZ\wedge NZ, NZ\wedge PZ, PZ\wedge NZ, PZ\wedge PZ\}$. Similarly, the rule in Example 2 has the input pattern as $\{N\wedge Z, NZ\wedge Z, Z\wedge Z, PZ\wedge Z, P\wedge Z\}$.

A conflict occurs in the rule base if there exist two rules which have overlapping input patterns but different linguistic consequences. We divide all the rules in a rule base into several clusters according to rule conclusions such that rules in the same cluster suggest the same linguistic output. The number of clusters in the rule base is equal to the number of output fuzzy sets defined. The formal description of the input pattern of a cluster is given in Definition 5.

Definition 5. The input pattern of a cluster C is a union of input patterns of the rules which belong to it:

$$IP(C)=\bigcup_{r\in C}IP(r).\quad(9)$$

It is clear that there is no disagreement inside any cluster. But between two different clusters there may exist conflict if the intersection between their input patterns is not empty. The size of this intersection set reflects the conflicting scale between two clusters.

Definition 6. The conflicting scale (CS) between two clusters C_1 *and* C_2 is the cardinality measure of the intersection set between their input patterns:

$$CS(C_1,C_2)=\|IP(C_1)\cap IP(C_2)\|.\quad(10)$$

Definition 7. The conflicting amount (CA) in the whole knowledge base is a sum of the conflicting

scales between any two distinct clusters in it:

$$CA = \sum_{i=1}^{N_{c-1}} \sum_{j=i+1}^{N_c} CS(C_i, C_j), \tag{11}$$

where N_c represents the number of clusters in the rule base.

Finally, we consider a numerical index to evaluate the consistency of the rule base. This index is designed to have its value to be in inverse proportion to the conflicting amount. In the case of no conflict existing, the consistency index reaches its maximal value of "1". Otherwise it decreases linearly with the increment of conflicting amount, until its minimal value "0" is reached. The calculation of consistency index λ_c is given in the following formula:

$$\lambda_c = \begin{cases} 1 - k\,CA & \text{if } k\,CA < 1, \\ 0 & \text{if } k\,CA \geqslant 1, \end{cases} \tag{12}$$

where $k \in (0, 1]$ denotes the decreasing rate of λ_c. The value of k should be determined in terms of particular problems to be solved.

To illustrate the calculation of consistency measure concretely, an example is given below:

Example 3.

r_1: IF $[x_1 = (N \text{ or } Z)]$ and $[x_2 = (N \text{ or } P)]$ THEN $y = Z$,

r_2: IF $[x_1 = (N \text{ or } Z)]$ THEN $y = N$,

r_3: IF $[x_2 = N]$ THEN $y = Z$,

r_4: IF $[x_1 = P]$ and $[x_2 = Z]$ THEN $y = P$,

r_5: IF $[x_1 = (Z \text{ or } P)]$ and $[x_2 = (Z \text{ or } P)]$ THEN $y = P$.

Consider a system with x_1, x_2 as its inputs and y as its output. Every input/output variable is characterised by three linguistic terms: *negative*(N), *zero*(Z) and *positive*(P). The knowledge base for the system contains the above five rules. The consistency index of it is derived as follows:

(1) The input patterns of the individual rules are constructed:

$$IP(r_1) = \{N \wedge N, N \wedge P, Z \wedge N, Z \wedge P\};$$

$$IP(r_2) = \{N \wedge N, N \wedge Z, N \wedge P, Z \wedge N, Z \wedge Z, Z \wedge P\};$$

$$IP(r_3) = \{N \wedge N, Z \wedge N, P \wedge N\};$$

$$IP(r_4) = \{P \wedge Z\};$$

$$IP(r_5) = \{Z \wedge Z, Z \wedge P, P \wedge Z, P \wedge P\}.$$

(2) According to the suggested conclusions, the rules are divided into three clusters:

$$C_1 = \{r_1, r_3\}, \quad C_2 = \{r_2\}, \quad C_3 = \{r_4, r_5\}.$$

(3) The input patterns of the clusters are obtained through union operation:

$$IP(C_1) = IP(r_1) \cup IP(r_3) = \{N \wedge N, N \wedge P, Z \wedge N, Z \wedge P, P \wedge N\};$$

$$IP(C_2) = IP(r_2) = \{N \wedge N, N \wedge Z, N \wedge P, Z \wedge N, Z \wedge Z, Z \wedge P\};$$

$$IP(C_3) = IP(r_4) \cup IP(r_5) = \{P \wedge Z, Z \wedge Z, Z \wedge P, P \wedge P\}.$$

(4) The conflicting scales between clusters are determined as follows:

$$IP(C_1) \cap IP(C_2) = \{N \wedge N, N \wedge P, Z \wedge N, Z \wedge P\};$$

$$CS(C_1, C_2) = \|IP(C_1) \cap IP(C_2)\| = 4;$$

$$IP(C_1) \cap IP(C_3) = \{Z \wedge P\};$$

$$CS(C_1, C_3) = \|IP(C_1) \cap IP(C_3)\| = 1;$$

$$IP(C_2) \cap IP(C_3) = \{Z \wedge Z, Z \wedge P\};$$

$$CS(C_2, C_3) = \|IP(C_2) \cap IP(C_3)\| = 2.$$

(5) Compute the conflicting amount and consistency index of the rule base:

$$CA = CS(C_1, C_2) + CS(C_1, C_3) + CS(C_2, C_3) = 7,$$

Set $k = 0.08$, then $\lambda_c = 1 - 0.08 \times 7 = 0.44$.

4. GA-based premise learning for fuzzy controller design

4.1. The basic introduction of genetic algorithms

GAs [2,5,8,9,20] are stochastic optimisation algorithms that emulate the mechanics of natural evolution. Based on probabilistic decisions, they exploit historic information to guide the search for new points in the problem space with expected improved performance. GAs are different from other traditional searching algorithms in the following aspects:

(1) GAs evaluate many points in the search space simultaneously, as opposed to a single point, thus reducing the chance of converging to the local optimum.
(2) GAs use only objective functions, therefore they do not require that the search space be differentiable or continuous.
(3) The genetic search is guided by probabilistic rules, rather than deterministic rules.

Essentially, a GA is an iterative procedure maintaining a constant population size. Any individual in the population encodes a possible solution to the considered problem into a string analogous to a chromosome in nature. At each iteration step new strings are created by applying genetic operators on selected parents, and subsequently some of the old weak strings are replaced by new strong ones. In this manner, the performance of the population will be gradually improved with proceedings of the evolutionary process.

We intend to employ a GA to learn premises of rules together with fuzzy set membership functions at the same time to acquire an optimal fuzzy controller. For this purpose, appropriate coding scheme, genetic operators and fitness function have to be defined.

4.2. Binary coding of rule premises

From Definition 1, we can see that the premise of a multiple-term unified fuzzy rule is characterised by sets $D(i) \subset \{1, 2, \ldots, q[p(i)]\}$. This fact suggests that a binary code be a suitable scheme for representing premises of such rules, as inclusion or exclusion of an integer in the sets $D(i)$ can be declared binary. For input variable x_j $(j = 1, 2, \ldots, n)$ with $q[j]$ linguistic terms, a segment consisting of $q[j]$ binary bits is required to encode the conditional composition for this variable. Every bit of the segment corresponds to a linguistic term with bit "1" for presence and bit "0" for absence of its fuzzy set in forming the condition. For example, assume that x_j has three linguistic terms $\{low, middle, high\}$, the segments "010" and "100" correspond to the conditions "$x_j = middle$" and "$x_j = low$" respectively. Similarly, the condition with OR-operation "$x_j = middle$ or $high$" is represented by the segment "011". All-one segment "111" is adopted in this paper to represent the wildcard of "do not care", meaning that the corresponding input variable is not considered in the rule premise. By composing the binary segments for individual inputs, the whole premise of a rule can be encoded into the group $P = (S(x_1), S(x_2), \ldots, S(x_n))$, where $S(x_j)$ denotes the segment of the condition description for input variable x_j.

Example 4. Assume four input variables x_1, x_2, x_3 and x_4 with $x_j = \{low, middle, high\}$ $(j = 1, \ldots, 4)$. The rule premise: $(x_1 = low$ or $high)$ and $(x_3 = middle)$ and $(x_4 = middle$ or $high)$ corresponds to the following binary coding:

$$S(x_1) = 101, \quad S(x_2) = 111, \quad S(x_3) = 010,$$
$$S(x_4) = 011,$$

$$P = [S(x_1), S(x_2), S(x_3), S(x_4)]$$
$$= [101\ 111\ 010\ 011].$$

It is worth noting that the following two cases lead to an invalid rule premise encoded:

(1) All the bits in the group are equal to one, meaning that all input variables are neglected in the premise.
(2) The group contains an all-zero segment. Its corresponding input variable thus takes no linguistic term in the premise, resulting in an empty fuzzy set for the condition of that input.

Rules with invalid premises are meaningless, play no role in the fuzzy reasoning. Therefore, they should

be removed from the knowledge base. This also provides the flexibility for the learning algorithm to adjust the size of the knowledge base.

Further, the coding of premise structure of a rule base is realised through merging binary groups of all individual rule premises in an head-to-tail manner. Let n_r be the maximal number of rules allowed to appear in the rule base (derived from the sum of upper limits of the rule numbers for individual output fuzzy sets), the binary code (CB) for rule premises as a whole is written as

$$CB = \{P(1), P(2), \ldots, P(n_r)\}, \tag{13}$$

where $P(i)$ $(i = 1, 2, \ldots, n_r)$ indicates the binary group for the premise of the ith rule. In cases of no invalid premises encoded, the corresponding rule base includes exactly n_r valid rules in it. Otherwise the size of the rule base can be reduced according to invalid premises detected in (13). For this reason, we claim that an adjustment of the size of the rule set is made possible within the limitation of prescribed maximal rule numbers for different rule conclusions.

4.3. Integer coding of membership functions of fuzzy sets

Triangular- and trapezoid-formed fuzzy sets are adopted in this paper. We impose some constraints on their membership functions to ensure that the sum of membership values for every variable is always equal to one, then certain parameters corresponding to endpoints (also peaks) of membership functions need to be tuned by GA. These parameters are critical for fuzzy partition of input/output variables, since they determine shapes and locations of fuzzy set membership functions.

Let the parameter for an endpoint be mapped by an integer N in the interval $\{0, 1, \ldots, N_{\max}\}$, the relationship between the parameter value V and the integer N is

$$V = V_{\min} + \frac{N}{N_{\max}}(V_{\max} - V_{\min}), \tag{14}$$

where $V_{\min}$ and $V_{\max}$ are two extreme limits of the endpoint under consideration. For fuzzy sets of a variable, there are usually several important endpoints which determine the distribution of their membership functions, and each of the endpoints can be quantised by an integer. The composition of such integers results in an integer-vector depicting fuzzy partition of that variable. For instance, consider the six fuzzy sets for a variable in Fig. 4. The outline of the membership functions of these six fuzzy sets are prescribed to satisfy the constraint mentioned above. Exact definition of membership functions is thus transformed to specification of positions of the four endpoints marked as circles in Fig. 4. If we quantise possible ranges of the positions from 0 to 100, the six fuzzy sets can be characterised by the integer-vector: $M = (15, 25, 45, 70)$. The elements in the vector M are ordered from small to big with correspondence to endpoints from left to right.

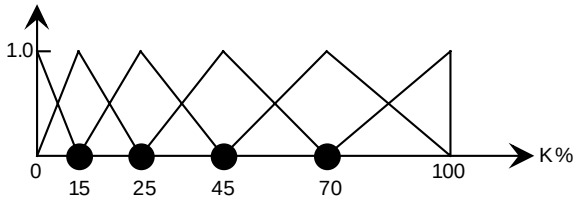

Fig. 4. Six fuzzy sets for a variable.

Further, the integer-vectors for membership functions of all individual variables are merged together to produce an integer-code to represent information of fuzzy sets as a whole. Suppose there are n_v input/output variables in the system and $M(i)$ $(i = 1, 2, \ldots, n_v)$ is the integer-vector for membership functions of the ith variable, the integer-code (CI) for the fuzzy sets of the whole control system is given as

$$CI = [M(1), M(2), \ldots, M(n_v)]. \tag{15}$$

4.4. Hybrid coding of a fuzzy controller

Because of the inherent relationship between fuzzy control rules and membership functions, both parts of the fuzzy controller should preferably be optimised simultaneously. For this purpose, a hybrid string consisting of a binary code in (13) and an integer-code in (15) as its two substrings is suggested, as illustrated in Fig. 5. The substring in the left is the binary code for premise structure of the rule base. n_r denotes the upper limit of the total number of rules in the rule base and $P(i)$ $(i = 1, \ldots, n_r)$ is the binary group corresponding to the premise of the ith rule. The substring

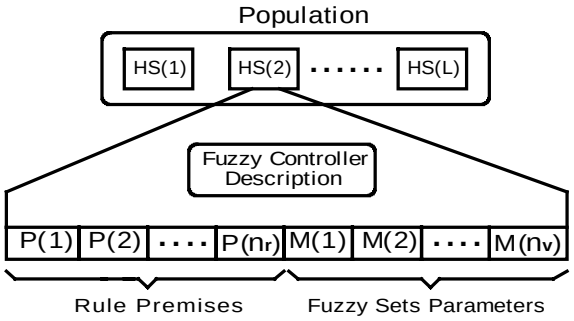

Fig. 5. A hybrid string in the population.

in the right is the integer-code representing information of fuzzy sets. $M(1), M(2), \ldots, M(n_v)$ indicate the integer-vectors for membership functions corresponding to variables $1, 2, \ldots, n_v$ respectively. n_v is the total number of input/output variables in the control system. Such a hybrid string forms an individual in the population. Each individual $HS(i)$ in the population is associated with a possible specification of the fuzzy controller to solve the problem. Through evolutionary process based on genetic operators, the quality of hybrid strings in the population can be gradually improved.

4.5. Fitness evaluation

Individual evaluation is a necessary step for the GA to search for optimal fuzzy controllers. To enable such an evaluation, every hybrid string of the population must be decoded into a fuzzy controller description, deleting rules with invalid premises at the same time. The GA searches in the problem space for solutions that satisfy two requirements: (1) the encoded fuzzy controller has good control performance; and (2) the rule base for the fuzzy controller has no or little inconsistency in it. The above two criteria should be combined in some manner to evaluate individuals of the population. Generally, the fitness function of an individual HS can be written as

$$Fit(HS) = com(f_P(HS), \lambda_c(HS)). \tag{16}$$

Here f_P is a numerical assessment of control performance. The definition of f_P is problem dependent. λ_c is the consistency index described in Section 3. The values $f_P(HS)$ and $\lambda_c(HS)$ are delivered to a mathematical expression com to obtain the fitness value of the individual HS.

4.6. Evolutionary process for fuzzy controller design

Individual strings are selected randomly to reproduce themselves based on the principle of "survive of the fittest". Each member in population is given a probability proportional to its fitness value for being selected as parent. The selected parents then undergo the genetic operators to produce their offspring. Since strong members in the population have more chances to reproduce themselves, while weak members have less chances and they become extinct with the time, the evolutionary process is made towards producing more fit members and eliminating less fit ones. Two important genetic operators are crossover and mutation.

Three-point crossover: By the operation of crossover, parent strings (old fuzzy controllers) mix and exchange their attributes through a random process, so that offspring (new fuzzy controllers) with even higher fitness than current individuals will be generated. Considering discrete premise structure and essentially continuous parameters of fuzzy sets, the two substrings in the chromosome encode information of different nature. It is preferable that the attributes in both substrings be mixed and exchanged separately. Here a three-point crossover is used. One breakpoint of this operation is fixed to be the splitting point between both substrings, and the other two breakpoints can be randomly selected within the two substrings, respectively. At breakpoints the parents bits are alternatively passed on to the offspring. This means that offspring get bits from one of the parents until a breakpoint is encountered, at which they switch and take bits from the other parent.

Example 5. Consider two strings in the following:

$$X_1 = (b_1^1, b_1^2, b_1^3, b_1^4, b_1^5, b_1^6, b_1^7, b_1^8, b_1^9, b_1^{10}, b_1^{11}, b_1^{12} \mid c_1^1, c_1^2, c_1^3, c_1^4, c_1^5, c_1^6),$$

$$X_2 = (b_2^1, b_2^2, b_2^3, b_2^4, b_2^5, b_2^6, b_2^7, b_2^8, b_2^9, b_2^{10}, b_2^{11}, b_2^{12} \mid c_2^1, c_2^2, c_2^3, c_2^4, c_2^5, c_2^6).$$

Both X_1 and X_2 consist of two substrings $(b_i^1, b_i^2, \ldots, b_i^{12}), (c_i^1, c_i^2, \cdots, c_i^6)$ $(i = 1, 2)$ representing rule premises and parameters of fuzzy sets, respectively. The position between $b_i^{12}$and c_i^1 is the splitting point between two substrings. Selecting the other two breakpoints for the crossover operator as the position between b_i^5, b_i^6 and the position between c_i^4, c_i^5, we obtain the children as follows:

$$Y_1 = (b_1^1, b_1^2, b_1^3, b_1^4, b_1^5, b_2^6, b_2^7, b_2^8, b_2^9, b_2^{10}, b_2^{11}, b_2^{12} \mid c_1^1, c_1^2, c_1^3, c_1^4, c_2^5, c_2^6),$$

$$Y_2 = (b_2^1, b_2^2, b_2^3, b_2^4, b_2^5, b_1^6, b_1^7, b_1^8, b_1^9, b_1^{10}, b_1^{11}, b_1^{12} \mid c_2^1, c_2^2, c_2^3, c_2^4, c_1^5, c_1^6).$$

Clearly this three-point crossover used here is equivalent to two one-point crossovers operating on both substrings separately.

Mutation: Mutation is a random alteration of a bit in a string so as to increase the variability of population. Because of the distinct substrings used, different mutation schemes are needed.

(1) Since the parameters of membership functions are essentially continuous, a small mutation with high probability is more meaningful. Therefore, it is so designed that each bit in the substrings for membership functions undergo a disturbance. The magnitude of this disturbance is determined by a Gaussian distribution function.

(2) For the binary substring representing rule premises, mutation is simply to inverse a bit, replace "1" with "0" and vice versa. Every bit in this substring undergoes a mutation with the probability of about 0.033.

At last, we summarise the GA for fuzzy controller design as follows:

Step 1: For running GA, set the parameters such as maximal generation number, population size, crossover and mutation rates, etc. Set the parameters $V_{\max}$ and $V_{\min}$ for integer coding in (14).

Step 2: Generate initial population $Pop(0)$ composed of randomly generated hybrid strings and evaluate their fitness values with (16).

Step 3: Select parents from current population $Pop(t)$ according to probability distribution based on fitness values of individuals.

Step 4: Apply genetic operators on the selected parents to produce a set of offspring.

Step 5: Evaluate the fitness values of the offspring with (16).

Step 6: Choose the best L individuals from the population $Pop(t)$ and the offspring to form the next generation $Pop(t+1)$. (L is the population size.)

Step 7: Terminate the search procedure if a satisfactory fuzzy controller is found or the maximal number of generations is reached. Otherwise go to Step 3.

5. A case study and results

5.1. Basic description of the plant

As a case study the proposed method was applied to the problem of balancing an inverted pendulum in the laboratory. The constitution of the pendulum system is depicted Fig. 6, where a pole is hinged to a motor-driven cart which moves on a rail to its right and left. The pole has only one degree of freedom to rotate about the hinge point. The model of the inverted pendulum is described by the following differential equations [1]:

$$\frac{d^2x}{dt^2} = \frac{1}{1 + [P^2/(M\Theta - P^2)]\sin^2\alpha} \times \left[\frac{\Theta}{N_{01}^2}Ku - \frac{F_r\Theta}{N_{01}^2}\frac{dx}{dt} - \frac{P^2 g}{N_{01}^2}\sin\alpha\cos\alpha + \frac{PC}{N_{01}^2}\frac{d\alpha}{dt}\cos\alpha + \frac{P\Theta}{N_{01}^2}\left(\frac{d\alpha}{dt}\right)^2\sin\alpha\right], \quad (17)$$

$$\frac{d^2\alpha}{dt^2} = \frac{1}{1 + [P^2/(M\Theta - P^2)]\sin^2\alpha} \times \left[\frac{PgM}{N_{01}^2}\sin\alpha - \frac{CM}{N_{01}^2}\frac{d\alpha}{dt} - \frac{P}{N_{01}^2}Ku\cos\alpha + \frac{F_rP}{N_{01}^2}\frac{dx}{dt}\cos\alpha - \frac{P^2}{N_{01}^2}\left(\frac{d\alpha}{dt}\right)^2\sin\alpha\cos\alpha\right]. \quad (18)$$

Here x and α are the cart's displacement and the pole's angular rotation, respectively. u denotes the voltage of the driving motor which produces a force on the

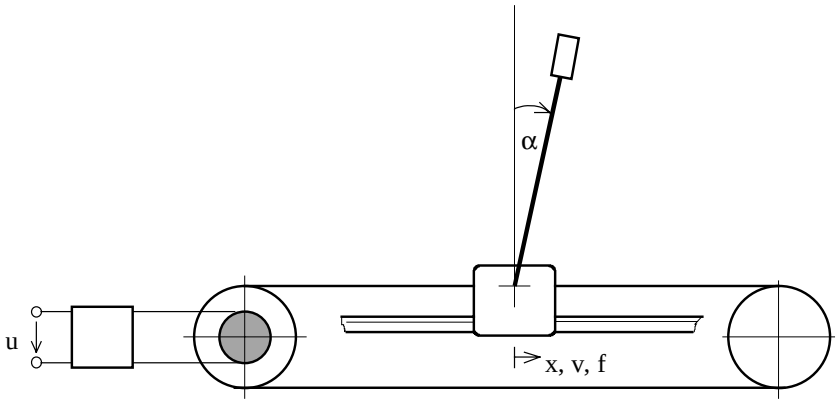

Fig. 6. The constitution of the pendulum system.

Table 1
Parameters of the inverted pendulum system

Parameter	Value	Unit
K	2.6	N/V
m_0	3.2	kg
m_1	0.329	kg
$M = m_0 + m_1$	3.529	kg
L_S	0.448	m
Θ	0.0743	kg m^2
$P = m_1 L_S$	0.1474	kg m
$N_{01}^2 = \Theta M - P^2$	0.2405	kg^2 m^2
Fr	6.2	kg/s
C	0.009	kg m^2/s

cart with a proportional magnitude to the value of u. The parameters of the considered pendulum system is listed in Table 1. It should be pointed out that this model is only an approximate description of the dynamics of the system. There are some factors such as the deadlock of the motor, non-uniformity of the friction and realistic noises which are not integrated into the differential equations. Later we will show that, in spite of the incompleteness and imprecision of the available model, a satisfactory fuzzy controller can be obtained with our method.

5.2. Fuzzy controller design based on GA

The objective is to control the driving forces of the motor so that the cart is positioned at the centre of the finite track while simultaneously balancing the pole hinged on the cart's top. The fuzzy controller for this purpose takes $x, \mathrm{d}x/\mathrm{d}t, \alpha$, and $\mathrm{d}\alpha/\mathrm{d}t$ as its input variables. The output of the controller is the voltage of the driving motor. Each of the inputs has three triangular or trapezoid fuzzy sets (*negative*, *zero* and *positive*), and five triangular or trapezoid fuzzy sets (NB, NS, Z, PS, PB) are used to partition the output space. The membership functions of the input and output fuzzy sets are depicted in Figs. 7 and 8, respectively. They are symmetric such that there are only six parameters to be tuned.

Based on the mathematical model of the plant, GA can be used to learn the fuzzy controller to balance the corresponding pendulum system. Failure is defined to occur when the angle α exceeds $[-12°, 12°]$ or the position x exceeds $[-0.4\ \mathrm{m},\ 0.4\ \mathrm{m}]$. We want not only to maximise survival time but also to minimise the discrepancy between the actual state (x, α) and the desired state $(x = 0, \alpha = 0)$ of the pendulum system. For evaluating the control behaviour, a set of different initial conditions should be introduced to cover a wide range of input space. The assessment $f_{\mathrm{P}}(HS) \in [0, 1]$ about the control performance is defined as follows:

$$f_{\mathrm{P}}(HS) = \frac{1}{13}\left\{\sum_{case\ 1}^{case\ 13}[s_n(1 - err_n)]\right\}, \tag{19}$$

where

$$s_n = \frac{s}{s_{\max}}, \quad err_n = \frac{1}{(2s)}\sum_{i=1}^{s}\left(\frac{|x_i|}{x_{\max}} + \frac{|\alpha_i|}{\alpha_{\max}}\right).$$

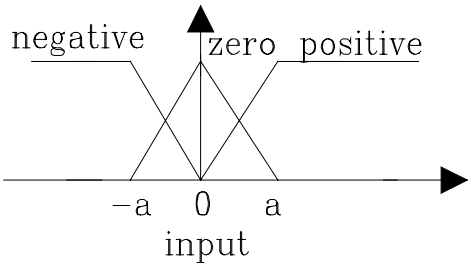

Fig. 7. Fuzzy sets for input variables.

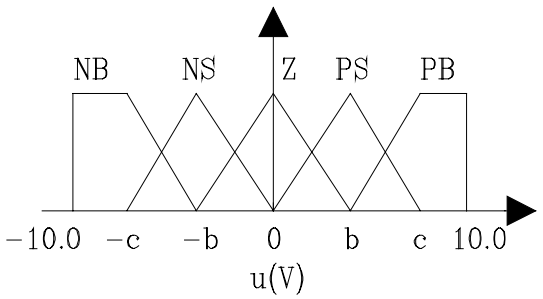

Fig. 8. Fuzzy sets for the output variable.

Here s is the number of sampling periods in the survival time of a trial, s_n and err_n are normalised survival time and normalised error, respectively. The total fitness value is the product of the control performance assessment defined in (19) and the consistency index of the rule base, thus we write

$$Fit(HS) = f_{\mathrm{P}}(HS) \times \lambda_{\mathrm{c}}(HS) \tag{20}$$

The upper limits of the numbers of rules for fuzzy actions NB, NS, Z, PS and PB are set in advance to be $5, 10, 4, 10$ and 5, respectively, meaning that totally 34 rules are supposed to be sufficient to achieve a desirable control performance. GA was used to learn premise structure of such possible rules and six parameters of the membership functions simultaneously by maximising the fitness function in (20). As a result of the genetic search, 22 rule premises were identified as invalid, therefore the rule base learned contains only 12 rules in it. Moreover, there is no linguistic conflict among the 12 rules in the knowledge base.

From this example, we clearly see that the exact rule number was not determined by hand in advance. What a human user needed to do is only a guess about the sufficient rule amounts for different conclusions.

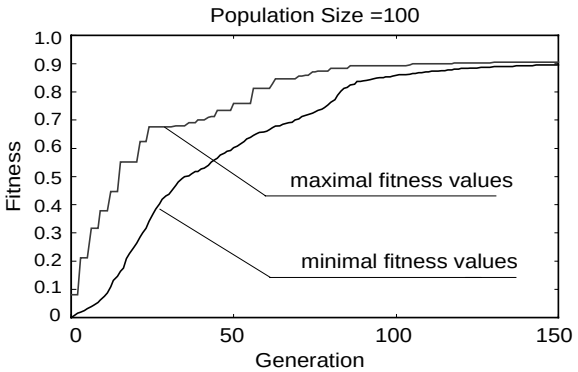

Fig. 9. Fitness values in generations.

GA can adapt the number of rules automatically with the possibility of obtaining a compact knowledge base whose size is much smaller than the specified upper limit.

Fig. 9 illustrates the evolution of the population during the running of the GA. The upper and lower curves correspond to the maximal and minimal fitness values, respectively, in each generation. In the experiments, the population size was selected as 100, mutation rate as 0.033, and the probability of crossover as 0.867.

5.3. Simulation results

Simulation tests reveal that the learned fuzzy controller not only successfully balances the pendulum of the exact model but also does well under some perturbations on the controlled system. As an example, we increase now the mass of the pole from 0.329 to 0.5 kg and the pole length form 0.448 to 0.57 m. Fig. 10 illustrates the controlled cart positions of the original and disturbed plants under the initial condition $(x = 0.16\ \mathrm{m}, x' = 0.05\ \mathrm{m/s}, \alpha = 6°, \alpha' = 7°/\mathrm{s})$, which is not included in the fitness function of the GA. Supposing the same initial state, the controlled pole angles for the original and disturbed plants are depicted in Fig. 11. From Figs. 10 and 11, it is easy to identify that the system changes stated above have only slight influences on control performance of the learned fuzzy controller.

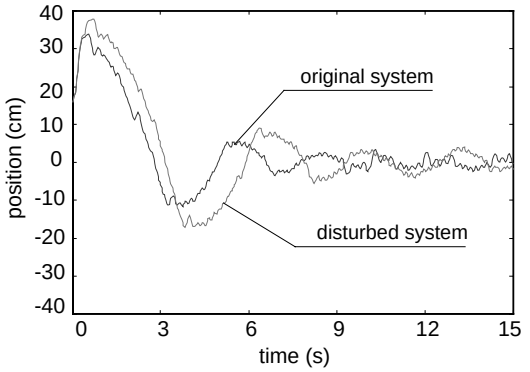

Fig. 10. The cart positions in the simulation.

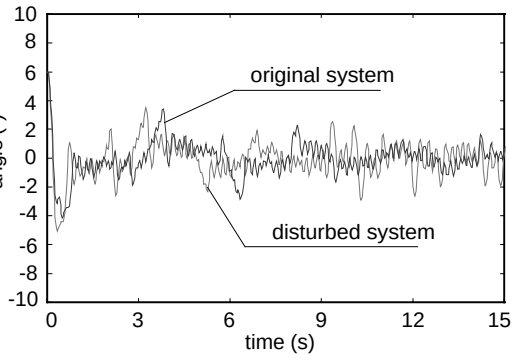

Fig. 11. The pole angles in the simulation.

5.4. Experimental results

The fuzzy controller learned also functions very well in real-time to balance the practical pendulum system in the laboratory, as shown in Fig. 12. The fuzzy control algorithm is implemented with a C-program running on PC. The available sensors provide information about position/velocity of the cart as well as angle of the pole. The angular velocity is derived through differentiation between two successive sampling instants. The sampling time of the control system is 30 ms. Controller's performance on the practical plant is depicted in Figs. 13 and 14, showing controlled cart positions and pole angles, respectively.

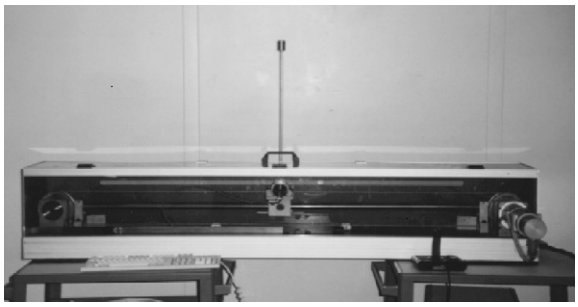

Fig. 12. The real pendulum system.

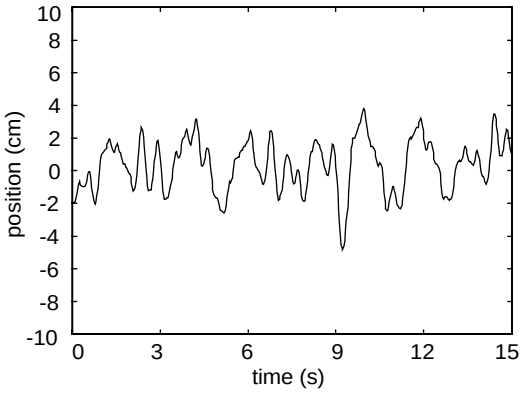

Fig. 13. The cart positions in the real operation.

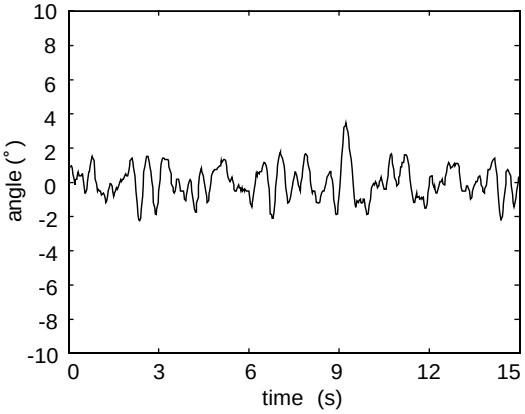

Fig. 14. The pole angles in the real operation.

5.5. Comparison with related works

GA-based fuzzy control for balancing an inverted pendulum was addressed by Varsek [26] and Karr [12] in the literature. They also used three linguistic terms to characterise each of the four input variables. Karr [12] applied a GA to adapt fuzzy membership functions for a fixed rule set. Varsek [26] employed GAs to derive control rules and tune parameters of membership functions in separate stages. A common feature of both works is that $3^4 = 81$ rules were required to enumerate every AND-combination of input fuzzy sets in the rule base. Our work presented here outsmarts those of Karr [12] and Varsek [26] in two aspects. First, the knowledge base learned in this paper has a much smaller size with only 12 rules, rather than 81 rules. The reason for this is that general premise structure of rules was allowed in our design procedure. Secondly, fuzzy rules as well as membership functions of our controller were optimised simultaneously. In addition, this paper presents more convincing success on a practical pendulum system, whereas Karr [12] and Varsek [26] only gave simulation results on simple plant models.

6. Conclusions

This paper proposes a new approach to fuzzy controller design by means of premise learning. It has the following important features:

(1) The rule premises are generalised via a GA rather than enumerated in terms of every canonical AND-connections of input fuzzy sets.
(2) A parsimonious rule base is made possible due to: (a) selection of important regions of the input domain to be covered; (b) general rule premises (multiple-value fuzzy logic and/or incomplete structure) allowed in the coding scheme.
(3) Linguistic conflicts in the rule base can be removed by incorporating the consistency index into the fitness function of the GA.
(4) A variable size of the rule base is possible in the sense that we do not have to specify the exact rule numbers for individual conclusions. What is needed is only a "guess" about how many rules appear sufficient to solve the problem under consideration.
(5) A simultaneous optimisation of control rules and membership functions of fuzzy sets is realised.

One drawback of the current work lies in the determination of the upper limits of control rules. When the upper limits are supposed to be too small, no satisfactory solution can be found. On the other side, a too "generous" estimation means a chromosome length much longer than necessary, so that the search space for the GA will become extremely large. Heuristic knowledge or experiences play a key role in making a favourable estimation about the sufficient amount of rules to achieve the control goal. Further research will be concentrated on developing a GA with variable length of chromosomes to enable automatic identification of the number of rules required without a priori information.

Acknowledgements

We would like to thank the anonymous referees sincerely for their helpful comments and suggestions.

References

[1] amira GmbH, Dokumentation zur Laborversuche invertiertes Pendels, Duisburg, 1992 (in German).
[2] S. Austin, An introduction to genetic algorithms, AI Expert. 1 (1990) 49–53.
[3] B. Carse, T.C. Fogarty, A. Munro, Evolving fuzzy rule based controllers using genetic algorithms, Fuzzy Sets and Systems 80 (1996) 273–293.
[4] H.J. Cho, Automatic rule generation for fuzzy controllers using genetic algorithms: a study on representation scheme and mutation rate, Proc. 7th IEEE Internat. Conf. on Fuzzy Systems, Anchorage, AK, 1998, pp. 1290–1295.
[5] L. Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.
[6] D. Driankov, H. Hellendoorn, M. Reinfrank, An Introduction to Fuzzy Control, 2nd Edition, Springer-Verlag, Berlin, 1996.
[7] D.S. Filipe, Supervision of fuzzy controllers using genetic algorithms, Proc. 7th IEEE Internat. Conf. on Fuzzy Systems, Anchorage, AK, 1998, pp. 1241–1246.
[8] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[9] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

[10] A. Homaifar, Ed McCormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms, IEEE Trans. Fuzzy Systems 3 (1995) 129–138.
[11] H. Inoue, Rule pairing methods for crossover in GA for automatic generation of fuzzy control rules, Proc. 7th IEEE Internat. Conf. on Fuzzy Systems, Anchorage, AK, 1998, pp. 1223–1228.
[12] C.L. Karr, Design of an adaptive fuzzy logic controller using a genetic algorithm, Proc. Internat. Conf. on Genetic Algorithms, San Mateo, CA, 1991, pp. 450–457.
[13] C.L. Karr, Genetic algorithm for fuzzy logic controller, AI Expert 2 (1991) 26–33.
[14] C.L. Karr, E.J. Gentry, Fuzzy control of pH using genetic algorithms, IEEE Trans. Fuzzy Systems 1 (1993) 46–53.
[15] J. Kinzel, F. Klawonn and R. Kruse, Modifications of genetic algorithms for designing and optimising fuzzy controllers, Proc. 1st IEEE Internat. Conf. on Evolutionary Computation, Orlando, FL, 1994, pp. 28–33.
[16] H. König, L. Litz, Inconsistence detection—a powerful means for the design of MIMO fuzzy controllers, Proc. 5th IEEE Conf. on Fuzzy Systems, New Orleans, LA, 1996, pp. 1191–1197.
[17] C.C. Lee, Fuzzy logic in control systems: Fuzzy logic controller—Part I, IEEE Trans. Systems Man Cybernet. 20 (1990) 404–418.
[18] M. Lee, H. Takagi, Integrating design stages of fuzzy systems using genetic algorithms, Proc. 2nd IEEE Internat. Conf. on Fuzzy Systems, San Francisco, CA, 1993, pp. 612–617.
[19] R.H. Li, Y. Zhang, Fuzzy logic controller based on genetic algorithms, Fuzzy Sets and Systems 83 (1996) 1–10.
[20] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 2nd Edition, Springer, Berlin, 1992.
[21] D.T. Pham, D. Karaboga, Optimum design of fuzzy logic controllers using genetic algorithms, J. Systems Eng. 1 (1991) 114–118.
[22] D. Popovic, N. Xiong, Design of flexible structured fuzzy controllers using genetic algorithms, Proc. IEEE Internat. Conf. on Fuzzy Systems, New Orleans, LA, 1996, pp. 1682–1686.
[23] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modelling and control, IEEE Trans. Systems Man Cybernet. 15 (1985) 116–131.
[24] Y.S. Tarng, Z.M. Yeh, C.Y. Nian, Genetic synthesis of fuzzy logic controllers in tuning, Fuzzy Sets and Systems 83 (1996) 301–310.
[25] P. Thrift, Fuzzy logic synthesis with genetic algorithms, Proc. Internat. Conf. on Genetic Algorithms, San Mateo, CA, 1991, pp. 509–513.
[26] A. Varsek, T. Urbancic, B. Filipic, Genetic algorithm in controller design and tuning, IEEE Trans. Systems Man Cybernet. 5 (1993) 1330–1339.
[27] N. Xiong, Fuzzy tuning of PID controllers based on genetic algorithms, Systems Sci. 25 (1999) 65–75.
[28] L. Zadeh, Outline of a new approach to the analysis of complex systems and design processes, IEEE Trans. Systems Man Cybernet. 3 (1973) 28–44.