The Third AFSS(1998), 646－651

FUZZY RULE MODIFICATION BY GENETIC ALGORITHMS

Seihwan Park Hyung Lee-Kwang

Dept. of Computer Science, KAIST(Korea Advanced Institute of Science and Technology)
373-1 Kusong-dong, Yusong-gu, Taejon, 305-701 KOREA
Tel:+82-42-869-3521 Fax:+82-42-869-9001 E-mail:{seihwan,khlee}@fuzzy.kaist.ac.kr

Abstract
Fuzzy control has been used successfully in many practical applications. In traditional methods, experience and control knowledge of human experts are needed to design fuzzy controllers. However, it takes much time and cost. In this paper, an automatic design method for fuzzy controllers using genetic algorithms is proposed. In the method, we proposed an effective encoding scheme and new genetic operators. The maximum number of linguistic terms is restricted to reduce the number of combinatorial fuzzy rules in the search space. The proposed genetic operators maintain the correspondency between membership functions and control rules. The proposed method is applied to a cart centering problem. The result of the experiment has been satisfactory compared with other design methods using genetic algorithms.

Keywords : Fuzzy logic controllers, Genetic algorithms, Encoding schemes, Crossover operator, Mutation operator

1. Introduction

Fuzzy theory has been developed by L. A. Zadeh in 1965. It has been used successfully in many control areas[1]. Fuzzy logic controllers(FLC's) are used when the processes are too complex for analysis by conventional mathematical techniques[14-17].

In traditional methods, experience and control knowledge of human experts are needed to design fuzzy controllers. However, it takes much time and cost. In addition, the traditional design process requires a lot of the trial-and-error, because the expert's knowledge is difficult to make concrete. Therefore, many automatic design methods have been proposed, e.g. fuzzy neural networks, fuzzy clustering methods, and gradient decent method[1,8]. The research based on GA's is now gaining interests[2-7,9-13].

In most automatic design methods for an FLC, the optimization of both the membership functions and the control rules of the FLC is required. The optimization is still dependent on the experience and knowledge of human experts[5-7,13]. That is, the ultimate associations among variables are derived from the knowledge of domain experts, who understand the semantics of the rules, Hence, either the linguistic terms or the fuzzy rules of an FLC are initially given by the expert's knowledge.

In this paper, we propose an automatic design method for fuzzy controllers using genetic algorithms. In the method, we proposed an effective encoding scheme, a new crossover operator, and a new mutation operator. In the proposed method, the maximum number of linguistic terms is restricted to reduce the number of combinatorial fuzzy rules in search space. When manipulating linguistic terms, the corresponding fuzzy rules are modified automatically. The proposed method is applied to the cart centering problem. Regarding the control speed and the probability of successful controls, the result of the experiment is satisfactory, compared with other design methods using genetic algorithms.

This paper is organized as follows. In Section 2, we explain the related works. In Section 3 the effective encoding scheme and the new genetic operators are proposed. In Section 4, we apply our method to the cart centering problem to demonstrate its performance. Finally, we conclude and discuss about further works.

2. Related Works

2.1 Fuzzy Logic Controllers(FLC's)

Fuzzy control is one of the most successful areas in the application of fuzzy theory. FLC's are excellent alternatives to the conventional control methodology when the processes are too complex for analysis by conventional mathematical techniques. An FLC consists of four components: a fuzzifier, an inference engine, a defuzzifier, and a knowledge-base. In this paper, the Mamdani's min-max operator in inferencing and the center of gravity as the defuzzification are used[1].

In many cases, the performance of FLC's depends on a designed knowledge-base in which membership functions and fuzzy control rules are defined. In traditional method, the knowledge-base is determined by experience and control knowledge of human experts. However, it is a trial-and-error process and takes much time and cost. Therefore, the automatic design method for FLC's, which can generate better both membership functions and control rules without human experts, is desirable.

2.2 Genetic Algorithms(GA's)

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. They can be considered as a general-purpose optimization method and have been successfully applied to search, optimization and machine learning tasks[4]. In GA's, a population of chromosomes are formed, each representing a possible solution to the problem. The population will undergo operations similar to genetic evolution, namely *reproduction*, *crossover* and *mutation*.

The following is a pseudo-code that describes a procedure of genetic algorithms.

```
Procedure genetic algorithm
{ P(t) : a population at generation t }
        t←0
        initialize P(t)
        evaluate P(t)
        while ( "not termination-condition" ) do
                t←t + 1
                select P(t) from P(t-1)
                alter P(t)
                evaluate P(t)
```

2.3 Other approaches to design FLC's using GA's

In the previous approaches using GA's, they applied GA's to tune the parameters of FLC's and fuzzy neural networks[8]. Recent approaches are to design FLC's directly, that is, to design membership functions and control rules. In these approaches, the one generated either membership functions or fuzzy rules[6,7,13], and the other designed both[5, 9,10].

Karr [6,7] used GA's to alter just the shape of fuzzy sets used in a given rule base. Thrift[13] used GA's to learn a rule table for given membership functions. Those approaches showed that GA's could reduce the development time and cost, and the designed FLC's outperformed human designed FLC's.

Takagi and Lee[10] determined membership functions and the number of fuzzy rules and the shape of fuzzy sets using GA's. Carse *et al*[3] used a real coded encoding scheme based on control rules which were composed of fuzzy numbers for input/output variables. They considered the relationship between membership functions and control rules because it was based on control rules.

As mentioned above, the previous works had some problems: (1) FLC's were to be still handled by human experts, (2)might increase the search space because of the long chromosomes, and (3) didn't consider the features of fuzzy controllers. To overcome those defects, we propose the method that can design both membership functions and control rules simultaneously. In the proposed method, we proposed an effective encoding scheme and new genetic operations considering the feature of FLC's. Our method can generate and modify membership functions and control rules.

3. Proposed Method

3.1 Encoding schemes

In order to design FLC's using GA's, encoding schemes by which an FLC is represented into a string(individual) are required. In this paper, FLC's are encoded in two parts: the part representing membership functions and the part representing control rules(rule table). For the purpose of easy explanations, we consider FLC's with two inputs *a* and *b*, and one output *c*, which are used frequently in practices.

3.1.1 Encoding scheme for membership functions

In this paper, linguistic terms in each variable are either triangular fuzzy numbers or trapezoidal fuzzy numbers in the both end sides of the universe of discourse. A triangular fuzzy number, which is the *i*th linguistic term, A_i, can be represented by the center position, A_i^{center} and the positions of both end sides on its base, A_i^{left} and A_i^{right}. The chromosome representing membership functions is coded in a binary string, consisting of two ordered sets of genes, *L* and *K* (see Fig. 1). The set *L* determines A_i^{center}'s, the center positions of triangular fuzzy number A_i's on the universe of discourse $[L_{left}, L_{right}]$. The set *K* represents A_i^{left} and A_i^{right} which are the positions of the both end sides on the base. The lengths of the set *L* and *K*, denoted by $|L|$ and $|K|$, mean the number of elements(genes) in set *L* and *K* respectively.

In the set *L*, a linguistic term A_i is determined by setting the value of gene to 1. The position of gene corresponds to the center position A_i^{center}. In the case of trapezoidal fuzzy numbers, all of genes corresponding to upper side of trapezoid are set to 1. The longer the length of the set *L* is, the higher the resolution presenting the positions on the universe of discourse is. When the universe of discourse is $[L_{left}, L_{right}]$, the resolution of the set *L* is $\frac{L_{right} - L_{left}}{|L|-1}$.

Using the resolution, we can calculate the center position, A_i^{center} of liguistic term A_i by Eq. (1).

$$A_i^{center} = L_{left} + \left(\frac{L_{right} - L_{left}}{|L|-1}\right) \times p, \quad p = 0,1,\cdots,|L|-1 \tag{1}$$

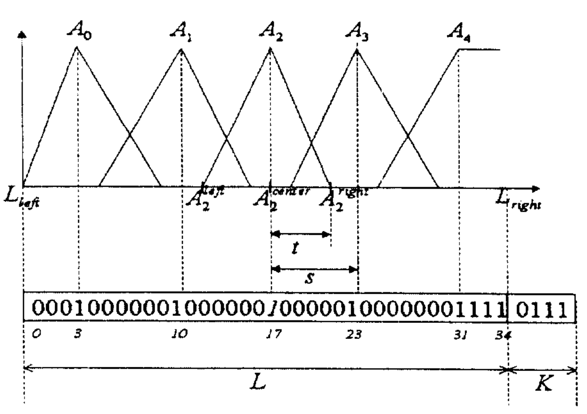


Fig. 1 Encoding for a membership function

Fig.1 shows an encoding example for membership functions in which the length of set *L* is 35, and the number of linguistic terms is 5.

The positions of both left and right end sides, A_i^{left} and A_i^{right} can be determined by ratio of two values, *s* and *t*. *s* is the distance between the center position A_i^{center} of the *i*th linguistic term and the center position A_{i+1}^{center} of the neighboring lingustic term A_{i+1}. *t* is the distance between A_i^{center} and the position of right end point, A_i^{right}.

In the proposed encoding schemes, the ratio *t/s* is determined by the set *K*,(0111 in Fig. 1). When the value of the *j*th gene in the set *K* is p_j, the ratio *t/s* is calculated by Eq. (2)

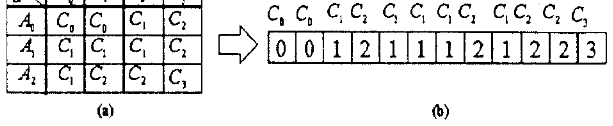
$$t/s = 0.5 + \frac{1}{2^{|K|+1}} \sum_{j=0}^{|K|-1} p_j 2^{|K|-j-1} \tag{2}$$

Then, using the ratio *t/s*, A_i^{left} and A_i^{right} can be calculated by Eq. (3).

$$\begin{aligned} A_i^{right} &= A_i^{center} + (A_{i+1}^{center} - A_i^{center}) \times t/s \\ A_i^{left} &= A_i^{center} - (A_i^{center} - A_{i-1}^{center}) \times t/s \end{aligned} \tag{3}$$

3.1.2 Encoding scheme for control rules

Generally, all of control rules combined by two input variables, are represented in the form of a rule table. In this paper, a chromosome representing control rules is formed by going row-wise in a rule table like Fig. 2. Each gene represents a control rule, that is, one cell in a rule table. The value of a gene is the value of *i*, the subscript of the *i*th linguistic term C_i in the output variable *c*. The length of a chromosome is variable, because each number of the linguistic terms in two input variables *a* and *b* is not fixed.



a \ b	B_0	B_1	B_2	B_3
A_0	C_0	C_0	C_1	C_2
A_1	C_1	C_1	C_1	C_2
A_2	C_1	C_2	C_2	C_3

(a)

C_0	C_0	C_1	C_2	C_1	C_1	C_1	C_2	C_1	C_2	C_2	C_3
0	0	1	2	1	1	1	2	1	2	2	3

(b)

Fig. 2 Encoding for control rules

For example, Fig. 2 shows that a rule table(a) is encoded into a chromosome 001211121223 (b).

3.1.3 Encoding scheme for fuzzy controllers

In order to encode fuzzy controllers, we integrate the proposed encoding schemes for membership functions and control rules, and encode in a string like Fig. 3. Then, an

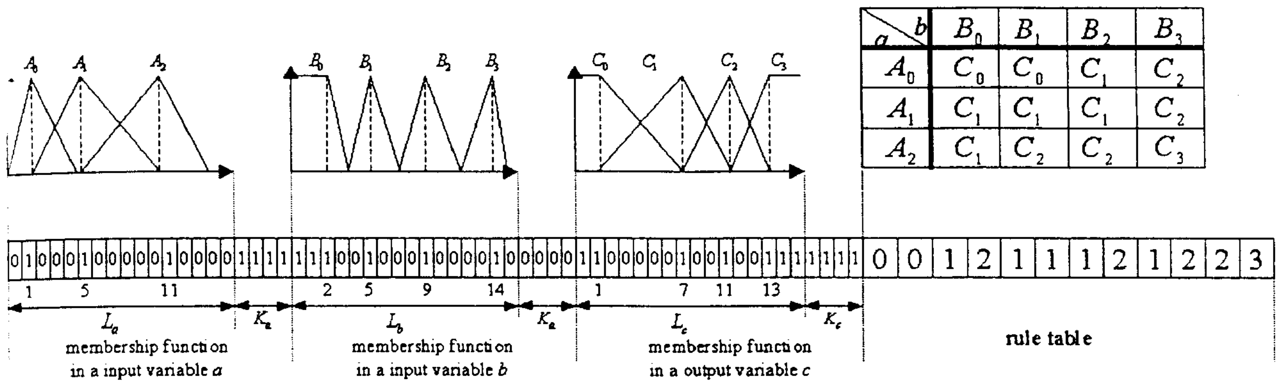

A_0 A_1 A_2
B_0 B_1 B_2 B_3
C_0 C_1 C_2 C_3
a b B_0 B_1 B_2 B_3
A_0 C_0 C_0 C_1 C_2
A_1 C_1 C_1 C_1 C_2
A_2 C_1 C_2 C_2 C_3
0 0 1 2 1 1 1 2 1 2 2 3
1 5 11
2 5 9 14
1 7 11 13
L_a K_a L_b K_a L_c K_c
membership function in a input variable a
membership function in a input variable b
membership function in a output variable c
rule table

Fig. 3 Encoding for a fuzzy controller

individual represents a fuzzy controller with membership functions for 3 variables and control rules.

The proposed encoding scheme has some advantages compared with real coded methods. One of them is to reduce the whole search space. But the resolution of our encoding schemes is lower than that of real coded methods. Although we can have high resolution as increasing the length of the set representing membership functions, as the results, the search space is increased accordingly. This is why the maximum number of linguistic terms depends on the length of chromosomes, and the number of control rules is a product of the number of linguistic terms in two input variables. In the proposed method, we limit the maximum number of linguistic terms in input-output variables under a reasonable number. Then, the maximum number of linguistic terms has no relation with the resolution(the length of the set L). The resolution can be higher without the increase of the search space.

The proposed encoding scheme cannot use the initialization process of a simple genetic algorithm[4]. The initialization process for our proposed encoding scheme is following.

First, it determines the number of linguistic terms in each variable. It must be less than the restricted number. Then in the chromosome presenting membership functions, the positions of the genes are selected randomly and the genes are set to 1 as many as the determined number. Set K, by which the base of linguistic terms is determined, is initialized and operated in the same way used in a simple genetic algorithm. After initializing membership functions, the chromosome representing control rules is initialized randomly. The value of genes in the chromosome is within the number of linguistic terms in output variable.

The initialized population will undergo the proposed crossover and mutation operations described in next.

3.2 Crossover

To maintain the correspondency between membership functions and control rules, we propose a new crossover operator. By this crossover operator, linguistic terms and control rules related with them are exchanged with each other. This prevents the loss of meaningful relationships between membership functions and control rules, and provides with a direction to generate a fuzzy controller with a good performance.

The followings are the notations used in the algorithm for crossover operation.

n_a, n_b, n_c :	the number of linguistic terms in variables a, b, c at the individual 1, respectively
n'_a, n'_b, n'_c :	the number of linguistic terms in variables a, b, c at the individual 2, respectively
A_i :	the i^{th} linguistic term in variable a at the individual 1
A'_i :	the i^{th} linguistic term in variable a at the individual 2
$L(0{:}l\text{-}1)$:	an array presenting the set of genes whose length is l
$L(i)$:	the value of the i^{th} gene in an array $L(0{:}l\text{-}1)$
$f(L, 0, l\text{-}1)$:	a function that returns the number of i satisfying $L(i)$=1 in an array $L(0{:}l\text{-}1)$
$h(L, j)$:	a function that returns the position of gene, corresponding to the j^{th} linguistic term, in an array $L(0{:}l\text{-}1)$
p :	the number of genes that are exchanged on L_a at the individual 1
q :	the number of genes that are exchanged on L'_a at the individual 2
$G(0{:}p\text{-}1)$:	an array representing the set of genes, G that are exchanged on L_a at the individual 1
$G'(0{:}q\text{-}1)$:	an array representing the set of genes, G' that are exchanged on L'_a at the individual 2
$R(0:n_a-1)(0:n_b-1)$:	two dimensional array representing the rule table in the individual 1
$R'(0:n'_a-1)(0:n'_b-1)$:	two dimensional array representing the rule table in the individual 2
$R(i)(j)$:	the value(subscript) of linguistic term in the consequent part of the rule "If a is A_i and b is B_j" in $R(0:n_a-1)(0:n_b-1)$
$R'(i)(j)$:	the value(subscript) of linguistic term in the consequent part of the rule "If a is A'_i and b is B'_j" in $R'(0:n'_a-1)(0:n'_b-1)$

Next, the algorithm for the crossover operation is described.

[Step 1] determine one variable of a, b, c. let a be selected.

[Step 2] determine n, the number of linguistic terms exchanged

[Step 3] exchange n linguistic terms $A_{n_a-n},\cdots,A_{n_a-1}$ in the individual 1 with n linguistic terms $A'_{n'_a-n},\cdots,A'_{n'_a-1}$ in the individual 2. p and q of genes, which are in the right most side of the set L_a and L'_a (see Fig. 4) at the individual 1 and 2, are exchanged. p and q are calculated by following equations:

$$p = |L_a| - h(L_a, n_a - n - 1) + 1, \qquad (4)$$
$$q = |L'_b| - h(L'_b, n_b - n - 1) + 1$$

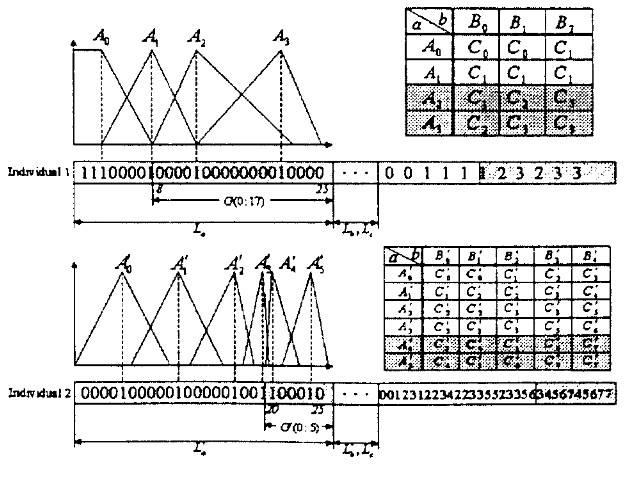
The following is the algorithm for the crossover for membership function.

```
Procedure CrossoverM F(G, G'', p, q)
Array : G(0 : p - 1), G'(0 : q - 1)
If p = q then
    for i from 0 to q - 1 do
        G'(i) ← G(i)
else If p < q then
   for i from 0 to q - 1 do
        G'(i) ← 0
    i ← 0
    while i < p
    do
        if G(i) = 1 then
            G'(⌊q / p × (i + 0.5)⌋ ← 1
            i ← i + 1
else (p > q)
    i ← 0
    right ← -1
    while i < q
   do
        left ← right + 1
        right ← p / q × (i + 1) - 1
        if f(G, left, right) = 0 then
            G'(i) ← 0
            i ← i + 1
        else
            k ← f(G, left, right)
            for j from 0 to k - 1 do
                G'(i + j) ← 1
            i ← i + k
```

[Step 4] When a input variable(a) is selected, control rules related to the linguistic terms participating in the crossover operation are changed. For a linguistic term $A_c(n_a - n \le c \le n_a - 1)$, $R(c)(0:n_b-1)$ presenting n_b control rules in the individual 1 are exchanged with $R'(c)(0:n_b-1)$ of n'_b control rules in the individual 2. According to the following algorithm, the control rules are exchanged respectively.

```
Array: $g(c)(0 : n_b n'_b - 1), g'(0 : n'_b - 1)$
for i from 0 to $n_b - 1$ do
    for j from 0 to $n'_b - 1$ do
        $g(n_b \cdot i + j) \leftarrow R(c)(i)$
for j from 0 to $n'_b - 1$ do
    $g'(j) \leftarrow \sum_{k=n_b j}^{n_b(j+1)-1} g(k) \Big/ n_b$
for j from 0 to $n'_b - 1$ do
    $R'(c)(j) \leftarrow \left\lceil \frac{n'_c}{n_c} \times g'(j) \right\rceil$
```

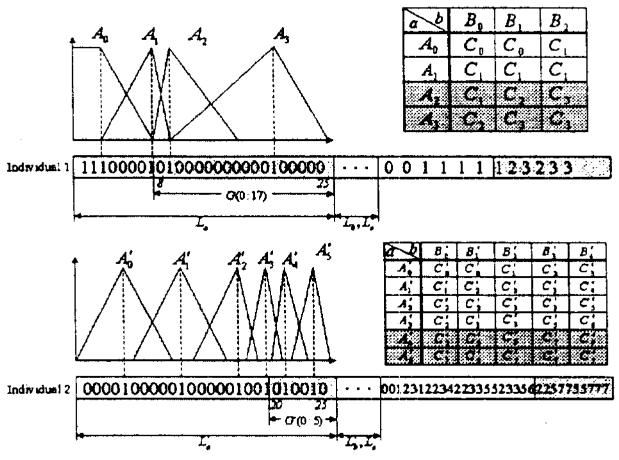
When a crossover operation is occurred in output variable c, the changes of control rules are not needed.



a \ b	B_0	B_1	B_2
A_0	C_0	C_0	C_1
A_1	C_1	C_1	C_1
A_2	C_2	C_2	C_2
A_3	C_2	C_3	C_3

a \ b	B'_0	B'_1	B'_2	B'_3	B'_4
A'_0	C'_0	C'_0	C'_1	C'_2	C'_3
A'_1	C'_1	C'_1	C'_2	C'_3	C'_3
A'_2	C'_3	C'_3	C'_3	C'_3	C'_3
A'_3	C'_3	C'_3	C'_3	C'_3	C'_4
A'_4	C'_3	C'_3	C'_4	C'_4	C'_4
A'_5	C'_4	C'_4	C'_4	C'_4	C'_4

Fig. 4 Before the crossover operation for membership functions



a \ b	B_0	B_1	B_2
A_0	C_0	C_0	C_1
A_1	C_1	C_1	C_1
A_2	C_2	C_2	C_2
A_3	C_2	C_3	C_3

a \ b	B'_0	B'_1	B'_2	B'_3	B'_4
A'_0	C'_0	C'_0	C'_1	C'_2	C'_3
A'_1	C'_1	C'_1	C'_2	C'_3	C'_3
A'_2	C'_3	C'_3	C'_3	C'_3	C'_3
A'_3	C'_3	C'_3	C'_3	C'_3	C'_4
A'_4	C'_3	C'_3	C'_4	C'_4	C'_4
A'_5	C'_4	C'_4	C'_4	C'_4	C'_4

Fig. 5 After the crossover operation for membership functions

Fig. 4 and Fig. 5 show before and after the crossover operation, which is occurred between the individual 1 and 2.

3.3 Mutation

3.3.1 Mutation for membership functions

When a mutation for membership functions is occurred, the value of gene representing the center position of a linguistic term is changed from 0 to 1 or from 1 to 0. According to change of the gene value, a new linguistic term is generated or an existing linguistic term is eliminated. To maintain the proposed encoding schemes, the number of linguistic terms after a mutation operation can not exceed over the restricted number.

When a mutation is occurred at the p^{th} gene in an input variable a, the linguistic term under the influence(generation or elimination) of the gene is assumed as the m^{th} linguistic term, A_m.

If a mutation is occurred between A_{m-1} and A_m, a new linguistic term becomes an A_m. The value of i in originally existing A_i is changed as follows.

$$i = \begin{cases} i + 1 & \text{if } i \ge m, \\ i & \text{otherwise} \end{cases}$$

If an existing linguistic term A_m is eliminated, the value of i in A_i is changed as follows.

$$i = \begin{cases} i & \text{if } i < m - 1, \\ i - 1 & \text{if } i > m \end{cases}$$

The mutation operation for membership functions causes the change of control rules related with the linguistic term A_m. When a mutation is occurred on the set L_a (see Fig. 6) in an input variable a, the rule table is modified(new control rules are generated or existing rules are eliminated).

The notations used in the algorithm are as follows:

n_a, n_b	: the number of linguistic terms in variable a and b before the mutation operation for membership functions
n'_a, n'_b	: the number of linguistic terms in variable a and b after the mutation operation for membership functions
p	: the position of gene in which a mutation is occurred
m	: the value(subscript) of the linguistic term influenced by $L(p)$
$L(0:l-1)$	: an array presenting the set of genes whose number is l
$R(0:n_a-1)(0:n_b-1)$	: two dimensional array representing the rule table before the mutation operation
$R'(0:n'_a-1)(0:n'_b-1)$	: two dimensional array representing the rule

table after the mutation operation

$R(i)(j)$: the value(subscript) of linguistic term in the consequent part of the rule "If a is A_i and b is B_j" at $R(0:n_a-1)(0:n_b-1)$

$R'(i)(j)$: the value(subscript) of linguistic term in the consequent part of the rule "If a is A'_i and b is B'_j" at $R'(0:n'_a-1)(0:n'_b-1)$

The following algorithm describes the procedure for the change of control rules

```
Array  $L(0:l-1), R(0:n_a-1)(0:n_b-1), R'(0:n'_a-1)(0:n'_b-1)$
If  $L(p)=1$ then
    $L(p) \leftarrow 0$
    $n'_a \leftarrow n_a - 1$
    $n'_b \leftarrow n_b$
    for $i$ from 0 to $n'_a$
        for $j$ from 0 to $n'_b$
            if $i<m$ then $R'(i)(j) \leftarrow R(i)(j)$
            else $R'(i)(j) \leftarrow R(i+1)(j)$
else
    $L(p) \leftarrow 1$
    $n'_a \leftarrow n_a + 1$
    $n'_b \leftarrow n_b$
    for $i$ from 0 to $n'_a$
        for $j$ from 0 to $n'_b$
            if $i<m$ then $R'(i)(j) \leftarrow R(i)(j)$
            else if $i=m$ then $R'(i)(j) \leftarrow \left\lfloor \frac{1}{2}(R(i-1)(j)+R(i)(j)) \right\rfloor$
            else $R'(i)(j) \leftarrow R(i-1)(j)$
```

When a mutation for membership functions in output variable c is occurred, the value of i's in linguistic terms C_i's is changed in the same manner as mentioned above. Then, existing C_i's in the original rule table are altered to the changed ones.

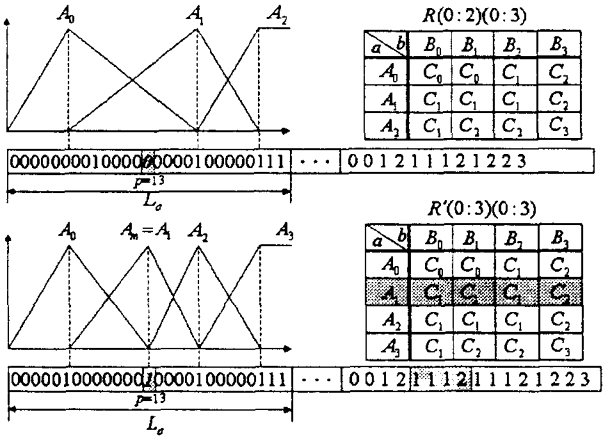

A_0 A_1 A_2
$R(0:2)(0:3)$
a b B_0 B_1 B_2 B_3
A_0 C_0 C_0 C_1 C_2
A_1 C_1 C_1 C_1 C_2
A_2 C_1 C_2 C_2 C_3
000000001000000000100000111
0 0 1 2 1 1 1 2 1 2 2 3
p=13
L_a
A_0 $A_m=A_1$ A_2 A_3
$R'(0:3)(0:3)$
a b B_0 B_1 B_2 B_3
A_0 C_0 C_0 C_1 C_2
A_1 C_1 C_1 C_1 C_2
A_2 C_1 C_1 C_1 C_2
A_3 C_1 C_2 C_2 C_3
000001000000001000001000000111
0 0 1 2 1 1 1 2 1 1 1 2 1 2 2 3
p=13
L_a

Fig. 6 mutation for membership functions

Fig. 6 shows that a mutation is occurred at the 13th gene on L_a and a new linguistic term A_1 and the related four rules are generated.

3.3.2 Mutation for a control rule

The value of genes representing control rules is determined by the value of i and the number of C_i's. The mutation for a control rule changes the value of gene, from i to j randomly. This means the consequent part of a control rule is changed from C_i to C_j, in which j is the subscript of other linguistic term $C_j (i \neq j)$ in an output variable c.

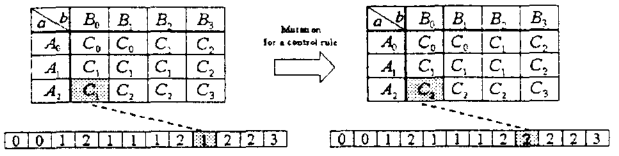

a b B_0 B_1 B_2 B_3
A_0 C_0 C_0 C_1 C_2
A_1 C_1 C_1 C_1 C_2
A_2 C_1 C_2 C_2 C_3
Mutation for a control rule
a b B_0 B_1 B_2 B_3
A_0 C_0 C_0 C_1 C_2
A_1 C_1 C_1 C_1 C_2
A_2 C_2 C_2 C_2 C_3
0 0 1 2 1 1 1 2 1 2 2 3
0 0 1 2 1 1 1 2 2 2 2 3

Fig. 7 mutation for a control rule

Fig. 7 shows that a mutation is occurred from 1 to 2 at the 9th control rule. It means that the consequent part of the control rule "If a is A_2 and b is B_0, then c is C_1", is changed to "c is C_2".

4. Experiments and Results

4.1 Cart centering problem

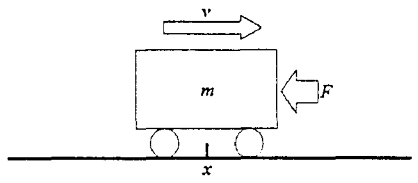


Fig. 8 Cart centering problem

In this experiment, a cart with mass m moves on one dimensional frictionless track(Fig. 8). The input variables for this problem are the location x and the velocity v of the cart. The output variable is the applied force, F. The objective is to find a fuzzy controller which can provide a force F which will move the cart to x=0 and v=0, from an arbitrary initial position and velocity in minimum time. The cart is simulated by the following equations of motion :

$$x(t+\tau)=x(t)+\tau v(t)$$
$$v(t+\tau)=v(t)+\tau\frac{F(t)}{m} \qquad (5)$$

To compare with other's experiments([3][13]), we choose time step τ=0.02sec and m=2.0kg. The range of applied force F is $-2.5N \le F \le 2.5N$.

4.2 Results

Fitness evaluation is identical to that used by Thrift[13] and Carse[3]. A simulation of the cart is run for 500 time step(10sec) with starting points (x_0, v_0) selected from 25 equally spaced positions in the range (-2.5,-2.5) to (2.5,2.5). The fitness is measured as (10-T) where T is the average time for the cart to reach the goal state such that max(|x|,|v|)<0.5. If more than 10sec of simulated time are required, it is considered as a failure of control and the fitness returned is 0.

The parameters for the genetic algorithm are as follows:

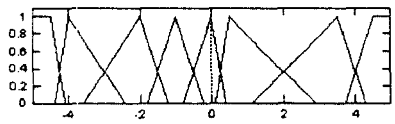
Size of population	: 100
Number of total generations	: 100
Probability of crossover	: 0.7
Probability of mutation	: 0.03
The length of set L	: 21
Max. number of linguistic terms	: 9

where the set L means a set of genes presenting membership functions in each variable.

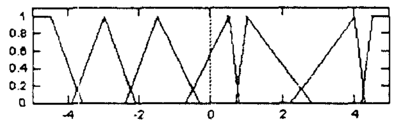
To compare the result of the proposed method with Thift[13] and Carse[3], we evaluated the best fuzzy controller with 100 random starting point in the range (-2.5,-2.5) to (2.5,2.5). We iterated an experiment 10 times. As the results, the cart was centered with about 98.4% of the rate of the successful controls and an average of 2.97sec. The following table represents the comparison for the average control time among the 3 methods.

Method	Thrift[13]	Carse[3]	Proposed
Average control time(sec)	3.28	2.90	2.97

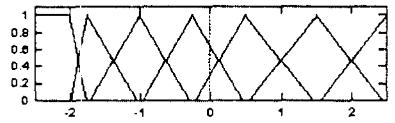
The proposed method generates membership functions and control rules simultaneously, compared with Thrift[13] which generated only control rules. Also, the performance of proposed method is better than that of Thrift and similar to that of Carse. However, our method designs membership functions in much understandable forms and covers the whole range of the universe of discourse. Fig. 9 and Fig. 10 show membership functions in each variable and a rule table designed by the proposed method, respectively.



(a) membership functions in a input variable v



(b) membership functions in a input variable x



(c) membership functions in a input variable F

Fig. 9 membership functions in the cart centering problem

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

Fig. 10 designed rule table

5. Conclusions

In this paper, we have proposed an automatic design method for fuzzy controllers using genetic algorithms. The proposed method can generate both membership functions and control rules simultaneously. In the proposed method, we also proposed the effective encoding scheme and new genetic operators. The proposed encoding scheme reduces the search space by eliminating the relationship between the resolution and the maximum number of linguistic terms. In the proposed genetic operations, the corresponding fuzzy rules are modified when manipulating linguistic terms. The proposed method was applied to the cart centering problem and showed that the control speed and the probability of successful controls are desirable.

For further works, fine tuning methods for the designed fuzzy controllers will be studied to increase their performance.

6. References

[1] H. Lee-Kwang, G. R. Oh, Fuzzy theory and applications I. II, Hongreung publishing co.(Korean), 1990.

[2] Seihwan Park, Y. I. Kim, J. K. Kim, and H. Lee-Kwang. An automatic design method for fuzzy controller using genetic algorithms. In Proc. of Fall Conf. on Korea Fuzzy and Intelligence Systems '96, pp. 306-309, 1996.

[3] Brian Carse, Terence C. Fogarty , and Alistair Munro. Evolving fuzzy rule based controllers using genetic algorithms. Fuzzy Sets and Systems, vol 80, pp. 273-293, 1996.

[4] David E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, 1989.

[5] Abdollah Homaifar and Ed McCormick. Simultaneous design of membership functions and rule sets for fuzzy controller using genetic algorithms. IEEE Trans. Fuzzy Systems, vol 3, no. 2, pp. 129-139, 1995.

[6] Charles L. Karr. Design of an adaptive fuzzy logic controller using genetic algorithms. In Proc. 4th Int. Conf. on Genetic Algorithms , pp. 450-457, 1991.

[7] Charles L. Karr and Edward J. Gentry . Fuzzy control of pH using genetic algorithms. IEEE Trans. Fuzzy Systems , vol 1, no. 1, pp. 46-53, 1993.

[8] K. M. Lee, D. H. Kwak, and H. Lee-Kwang, Fuzzy Inference Neural Network for Fuzzy Model Tuning, IEEE Trans. SMC, vol. 26, no. 4, pp. 637-645, 1996.

[9] J. H. Lee and H. Lee-Kwang, Fuzzy Identification of Unknown Systems based on GA, Lecture Notes in Artificial Intelligence 1285, Springer-Verlag, pp. 216-223, 1997.

[10] Michael A. Lee and Hideyuki Takagi. Integrating design stages of fuzzy systems using genetic algorithms. In Proc. 3rd IEEE Int. Conf. on Fuzzy Systems , pp. 612-617, 1993.

[11] Tadahiko Murata and Hisao Ishibuchi. Adjusting membership functions of fuzzy classification rules by genetic algorithms. In Proc. 5th IEEE Int. Conf. on Fuzzy Systems , pp. 1819-1824, 1995.

[12] Gregory V. TAN and Xiheng HU. On designing fuzzy controllers using genetic algorithms. In Proc. 6th IEEE Int. Conf. on Fuzzy Systems , pp. 905-911, 1996.

[13] Philip Thrift. Fuzzy logic synthesis with genetic algorithms. In Proc. 4th Int. Conf. on Genetic Algorithms, pp. 509-513, 1991.

[14] H. Lee-Kwang, and K. M. Lee. Fuzzy Hypergraph and Fuzzy Partition, IEEE Trans. Systems, Man, and Cybernetics, vol. 25, no. 1, pp. 196-201, 1995.

[15] C. B. Kim, K. A. Seong, H. Lee-Kwang. and J. O. Kim, Design and Implementation of FEGCS, IEEE Trans. Systems, Man, and Cybernetics, vol. 28 , part A, no. 3, 1998.

[16] Y. D. Kim and H. Lee-Kwang. High Speed Flexible Fuzzy Hardware for Fuzzy Information Processing, IEEE Trans. Systems, Man, and Cybernetics, vol. 27, no. 1, pp. 45-56, 1997.

[17] H. Lee-Kwang, K. A. Seong, and K. M. Lee. Hierarchical Partition of Nonstructured Concurrent Systems, IEEE Trans. Systems, Man, and Cybernetics, vol. 27, no. 1, pp. 105-108, 1997.