ELSEVIER

Fuzzy Sets and Systems 97 (1998) 1–7

FUZZY sets and systems

Genetic algorithms for learning the rule base of fuzzy logic controller

T.C. Chin*, X.M. Qi

School of Electrical & Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798, Singapore

Received October 1995; revised October 1996

Abstract

In this paper, genetic algorithms are used in the study to maximise the performance of a fuzzy logic controller through the search of a subset of rule from a given knowledge base to achieve the goal of minimising the number of rules required. Comparisons are made between systems utilising reduced rules and original rules to verify the outputs. As an example of non-linear system, an inverted pendulum will be controlled by minimum rules to illustrate the performance and applicability of this proposed method.

Keywords: Fuzzy logic control; Genetic algorithms; Rule base; Inverted pendulum

1. Introduction

Fuzzy logic controllers (FLCs) are being used successfully in an increasing number of application areas in the control community. FLCs are rule-based systems that use fuzzy linguistic variables to model human rule-of-thumb approaches to problem solving, and thus overcome the limitation that classical expert systems may meet because of their inflexible representation of human decision making. The major strength of fuzzy controllers also lies in the way a non-linear output mapping of a number of inputs can be specified easily using fuzzy linguistic variables and fuzzy rules.

When defining IF-THEN rules for the knowledge-base of a controller, we may meet some practical problems. First, because of hardware limitations such as memory, speed requirement, and cost, the whole set of rules is impractical in a complex system which requires a multi-input and multi-output controller. Second, an obvious problem in defining the knowledge-base for the controller is that no one can be sure that the rules are defined correctly and with no conflict with each other in certain situations. Usually, these inadequately defined rules or conflicted rules are very difficult to identify because of the complex controller along with the complex plant. Finally, in a human expert control case, a man may just use a small number of rules in his mind to control a complex system, apparently through the use of an incomplete but robust control protocol. So the ability to select an optimised subset of rules without degrading the controller's performance is a challenge in designing a fuzzy control system.

* Corresponding author. E-mail: etcchin@ntuvax.ntu.ac.sg.

0165-0114/98/$19.00
PII S0165-0114(96)00354-5

Genetic algorithms (GAs) are search algorithms based on the mechanics of natural selection and natural genetics. GAs have the properties that make them a powerful technique for selecting high-performance parameters for FLCs. For example, Lee and Takagi [3] design a fuzzy system using a genetic algorithm for the inverted pendulum. Karr and Gentry [2] control the pH of the acid–base system with fuzzy system's input membership functions manipulated by a GA. Park et al. [4] optimise a fuzzy reasoning model by genetic algorithm to control a direct current series motor.

Previous works focus only on optimising the FLC parameters and on reducing the number of the rules, while in this paper, genetic algorithms are used to search for an optimised subset of rules (both number of rules and the rule value) from a given knowledge-base to achieve the goal of minimising the number of rules used while maintaining the controller's performance. GA will eliminate all unnecessary rules, which have no significant contribution to improve the system performance, and also optimise the rule value from a controller's knowledge-base. The badly defined and conflicting rules are also eliminated through the GA search because their existence degrades the system performance. Therefore, genetic algorithms can be employed to help the design of the optimal fuzzy controller for a given plant system in terms of using a minimum number of rules.

In this paper the robustness and simple mechanics of GAs have been descriptively applied to establish the aforestated controller for a typical non-linear system-inverted pendulum.

The remainder of this paper is organised as follows. In Section 2, we give the review of fuzzy logic control, genetic algorithms. In Section 3, we demonstrate the implementation of GAs for the tuning of the fuzzy controller, and the effectiveness of the controller based on an intensive simulation study.

2. Genetic algorithm-based FLCs

2.1. Fuzzy logic controller (FLC)

Recall that the basic set of rules that capture the knowledge-based control algorithm called the fuzzy logic controller (FLC) is

$$IF\ U_1\ is\ B_{i1}\ AND\ U_2\ is\ B_{i2}\ THEN\ V\ is\ D_i, \tag{1}$$

where the inputs U_1, U_2 and output V are usually the error $e(k)$, its first difference $\Delta e(k)$, and the first difference of the control $\Delta u(k)$, respectively; B_{i1}, B_{i2} and D_i, $i = (1,\ldots,m)$ are linguistic values presented as fuzzy subsets of the universes X_1, X_2 and Y, respectively. The output of the FLC is (COA method)

$$y^* = \frac{\sum_{j=1}^{n} F(y_j) y_j}{\sum_{j=1}^{n} F(y_j)}, \tag{2}$$

where F is the fuzzy output inferred by the FLC

$$F(y) = \bigvee_{i=1}^{m} F_i(y), \tag{3}$$

where

$$F_i(y) = \tau_i \wedge D_i(y), \quad i = (1,\ldots,m) \tag{4}$$

and

$$\tau_i = B_{i1}(e(k)) \wedge B_{i2}(\Delta e(k)) \tag{5}$$

are the fuzzy output and the degree of firing (DOF) of the ith rule, obtained for given crisp inputs $U_1 = e(k)$ and $U_2 = \Delta e(k)$. Alternatively, by the simplified method of reasoning, the output inferred by the FLC is

$$y^* = \frac{\sum_{i=1}^{m} \tau_i y_i^*}{\sum_{i=1}^{m} \tau_i}, \tag{6}$$

where y_i^*'s are the defuzzified values (centroids) of the consequent of the rule

$$y_i^* = \frac{\sum_{j=1}^{n} D_i(y_j) y_j}{\sum_{j=1}^{n} D_i(y_j)}, \quad i = (1,\ldots,m). \tag{7}$$

2.2. Genetic algorithms: Mechanism and implementation

GAs are search algorithms based on the mechanics of natural selection and natural genetics. Unlike many classical optimisation techniques, genetic algorithms do not rely on computing local derivatives to guide the search process. Genetic algorithms also include random elements, which help to avoid getting trapped in local minima [1].

GAs generally consist of three fundamental operators: reproduction, crossover and mutation. Given an optimisation problem, simple GAs encode the parameters concerned into finite bit strings, and then run iteratively using the three operators in a random way but based on the fitness function evolution to perform the basic tasks of copying strings, exchanging portions of strings as well as changing some bits of strings, and finally find and decode the solutions to the processing of a GA, the following paragraphs explains the details:

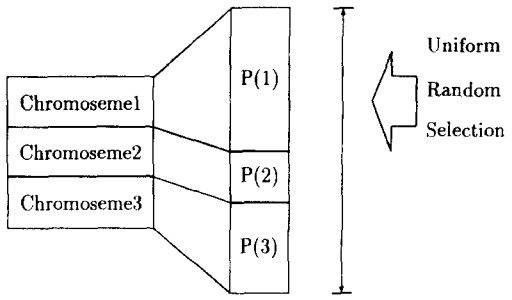
(1) *Coding the parameters.* It has become a common way to translate the parameters into binary bit strings. Several parameters are coded into one long string.

(2) *Initial generation.* It always begins by randomly generating an initial population of N strings, each of length m bits. The population size N is a compromising factor. Large N increases the possibility of including the solution in the first few generations, but decreases the running speed of the GAs mentioned above, the string length m determines the resolution. It has long been recognised that because genetic algorithms perform a global search of a solution space, the encoded bit-string lengths should be kept as short as possible, since the size of the search space increases exponentially with string size.

(3) *Fitness evaluation.* In the current generation, each of the strings is decoded to be its corresponding actual parameter. Then, these parameters are sent to a judgement machine which yields a measure of the solution's quality, evaluated with some objective functions and assigned individually with fitness values.

(4) *Reproduction.* Reproduction is a process by which the strings with larger fitness values can produce accordingly with higher probabilities large number of their copies in the new generation. The most common method used is the weighted roulette selection shown in the Fig. 1.

(5) *Crossover.* Crossover is a process by which the systematic information exchange between two strings is implemented using probabilistic decisions. In a crossover process, two newly reproduced strings are chosen from the mating pool and arranged to exchange their corresponding portions of binary strings at a randomly selected partitioning position along them. This process can combine better qualities among the preferred good strings. Uniform crossover does not select a set of crossover points. It simply considers each bit position of the two parents, and swaps the two bits with a probability of 50%.



For Relative Fitness:

f(Chrom1)>f(Chrom3)>f(Chrom2)

Fig. 1. Weighted roulette selection.

Crossover example.

Before crossover

$$\begin{aligned} A &= \overline{110111101} \mid 11011101011 \\ B &= 000100010 \mid \underline{00010010010} \end{aligned} \tag{8}$$

After crossover

$$\begin{aligned} A &= \overline{110111101} \mid 00011010010 \\ B &= 000100010 \mid \underline{11011101011} \end{aligned} \tag{9}$$

(6) *Mutation.* Mutation is a process by which the chance for the GA to reach the optimal point is reinforced through just an occasional alteration of a value at a randomly selected bit position. The mutation process may quickly generate those strings which might not be conveniently produced by the previous reproduction and crossover processes. Mutation may suddenly spoil the opportunity of the current appropriate generation, so, this process usually occurs with a small probability and is complementary to reproduction and crossover. In Eq. (9), the underline bit is changed from 0 to 1 because of mutation.

(7) *Iteration.* The GA runs iteratively repeating the processes (3)–(7) until it arrives at a predetermined ending condition. The speed of iteration depends not only on the population size N and the string length m but also has something to do with the selection of probabilities. Finally, the acceptable solution is obtained and decoded into its original pattern from the resulting binary strings.

Due to their properties, GAs differ fundamentally from the conventional search techniques. For example, they

- consider a population of points, not a single point,
- work directly with strings of characters representing the parameter set, not the parameters themselves,
- use probabilistic rules to guide their search, not deterministic rules,
- GAs require only information concerning the quality of the solution produced by each parameter set; this differs from many optimisation methods which require derivative information or in some cases, complete knowledge of the problem structure and parameters.

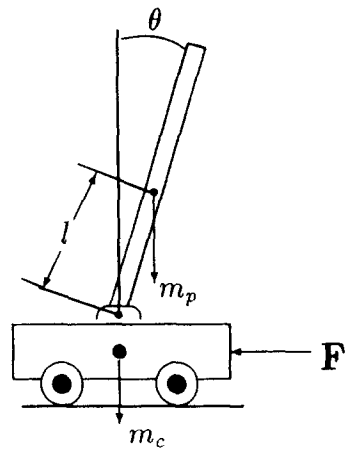


Fig. 2. Inverted pendulum.

3. Illustrative examples

3.1. Encode

To incorporate genetic algorithms into our scheme, we encoded the rule base into a long binary string. The coding is illustrated as

$$\overbrace{1\ \underbrace{010}_{\text{rule value}}}^{\text{rule1}}\ \overbrace{0\ \underbrace{110}_{\text{rule value}}}^{\text{rule2}}\ \cdots \tag{10}$$

There are four bits to represent one rule. The first bit is used to indicate whether the rule is used or not. That is, if this bit is 1, then the rule is survived, otherwise, abandoned. The next three bits represent the rule value which takes in the interval 0(000) to 7(111). While 0 means NB, 1 means NM, ..., 7 means PB.

In our simulations, the symmetrical Gaussian membership functions is used for fuzzification and defuzzification:

$$B_{ij}(x_j) = \exp\left(-\frac{1}{2}\left(\frac{x_j - x_{ij}^*}{\sigma_{ij}}\right)^2\right) \tag{11}$$

with parameter x_{ij}^* and σ_{ij}. The Gaussian form is chosen on the basis of formal considerations; membership functions of any other form, for example, triangular or trapezoidal, can be considered as well.

3.2. Performance criteria

The parameters for the GAs simulation are set as follows:

(1) Initial population size: 50;
(2) Maxnumber of generation: 100;
(3) Crossover: Uniform crossover with probability 0.8;
(4) Mutation probability: 0.01.

In this paper, the performance is measured using the following criteria.

(1) Minimum time-weighted integral of squared errors,

$$\text{TWISE} = \int_0^t t^{k_1} e^2\, dt, \quad k_1 = 0, 1, 2, \ldots, n; \tag{12}$$

(2) Combined performance index using overshoot (T_o) and rise time (T_r),

$$\text{CPI} = k_2 T_o + k_3 T_r, \tag{13}$$

where in Eqs. (12) and (13), k_1, k_2, k_3 are experimental parameters in order to emphasise our requirement about T_o or T_r.

3.3. Inverted pendulum example

For experimental purposes, we applied our method to the inverted pendulum problem. The goal is to balance the pole in the vertical position by applying an appropriate horizontal force to the cart. This example is only concerned with the control of the angular position of the pole and not the position or velocity of the cart. The cart travels in one direction along a frictionless track; see Fig. 2.

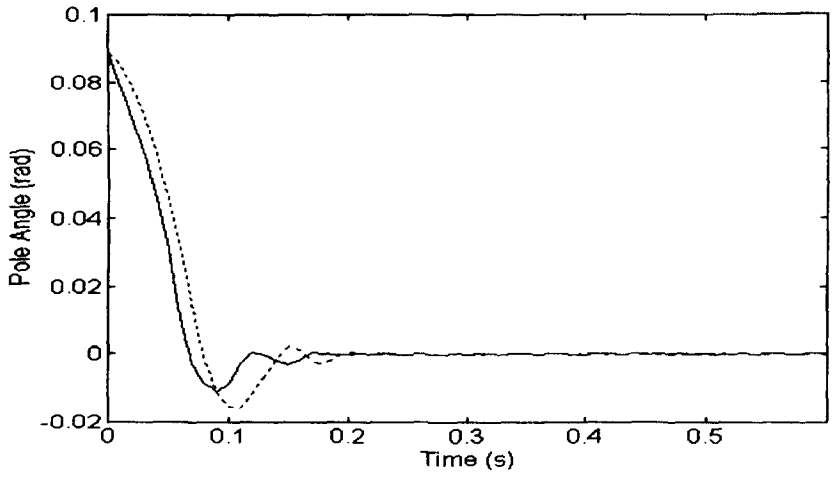


Fig. 3. Trajectory plot of integrated design (Test 1).

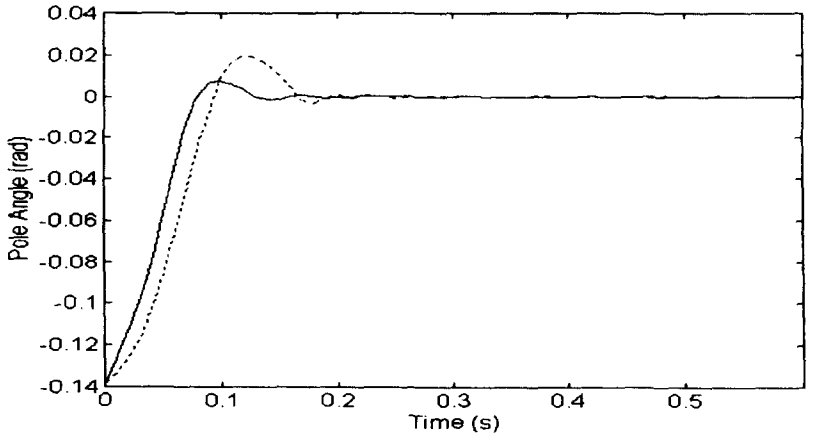


Fig. 4. Trajectory plot of integrated design (Test 2).

The definitions of the process parameters are as follows: θ is the angular position of the pole with respect to the vertical, F is the driving force on the cart, $2l$ is the length of the pole, m_p is the mass of the pole, m_c is the mass of the cart and g is the gravitational acceleration constant. The state equation describing only the angle dynamics of the inverted pendulum is given by

$$\dot{\theta} = \frac{d\theta}{dt}, \tag{14}$$

$$\ddot{\theta} = \frac{g\sin(\theta) - (\cos(\theta)/m_p + m_c)(m_p l\dot{\theta}^2 \sin(\theta) + F(t))}{(4l/3) - (m_p l\cos^2(\theta)/m_p + m_c)}, \tag{15}$$

where g is 9.8 m/s^2, $m_p = 0.1$ kg, $m_c = 0.9$ kg and $l = 0.5$ m.

In the simulation, the equations of motion are defined by differential equations. A four-step forward Runge–Kutta can be used to approximate its state at $t + h$.

3.4. Results and discussion

The simulation result is shown in Figs. 3–5 with the initial conditions set as in Table 1.

Figs. 3–5 show the superiority of the selected rules controller over the full rules ones. The solid

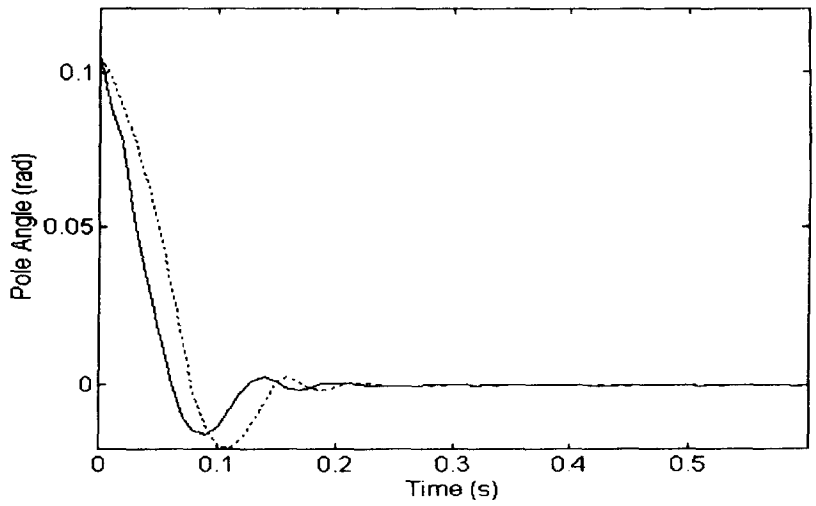


Fig. 5. Trajectory plot of integrated design (Test 3).

Table 1
The initial conditions of pendulum

Index	Test 1	Test 2	Test 3
θ	0.09	−0.14	0.108
$\dot{\theta}$	0.12	−0.02	−1.25

Table 2
Rule base before optimisation

$\Delta e/e$	NB	NM	NS	NZ	PZ	PS	PM	PB
PB	+3	+3	+3	+2	+2	+1	+1	0
PM	+3	+3	+2	+2	+1	+1	0	−1
PS	+3	+2	+1	+1	+1	+1	−1	−1
ZE	+2	+1	+1	0	0	−1	−1	−2
NS	+1	+1	0	−1	−1	−1	−3	−3
NM	+1	0	−1	−1	−3	−2	−3	−3
NB	0	−1	−2	−2	−3	−3	−3	−3

curves represent the integrated optimised results, the dashed curves are the rule selection optimised results.

Tables 2 and 3 are rule matrices before and after the integrated optimisation of the initial condition of 'Test 1', respectively. For 'Test 2' and 'Test 3', the results are similar. From Tables 2 and 3, we can see that the fuzzy controller with 43 rules instead

Table 3
Rule base after optimisation

$\Delta e/e$	NB	NM	NS	NZ	PZ	PS	PM	PB
PB		+3	+3			+2		0
PM	+3	+2	+2	+2	+1	+2	0	−1
PS			+1	+1	+1	+1	−1	−2
ZE	+2	+1		0	0		−1	−2
NS		+1	0	−1	−1	−1	−3	
NM	+1	0	−1	−1		−2	−2	−3
NB	0		−2		−3	−3	−3	−3

of 56 rules controls the inverted pendulum with better performance. The blank entry in the rule matrix means that particular rule contributes nothing to the overall performance of the controller. The FLC output at that entry is determined by other rules' interaction.

The control surface is shown in Figs. 6–9. Fig. 6 is the control surface without any optimisation. Fig. 7 corresponds to the initial condition of $\theta = 0.09, \dot{\theta} = 0.12$, while Fig. 8 and Fig. 9 are other two initial conditions, respectively. It is obvious that the original and optimised output surfaces are different. The control surface of original FLC is a general one which has no special characters because a symmetry control action is assumed. While the control surface of the optimised rule base has a wider control zone near the setpoint

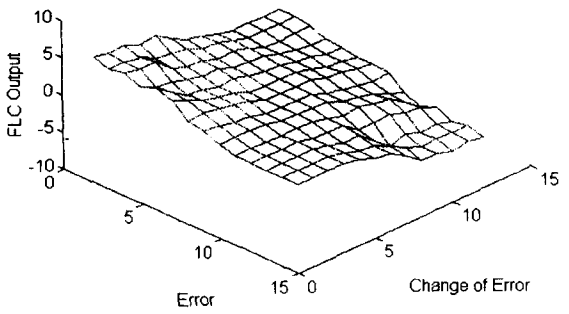


Fig. 6. Control surface of original FLC.

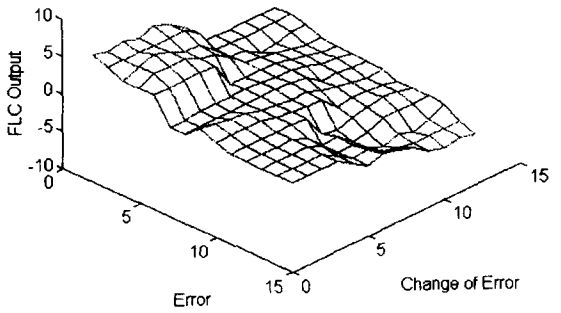


Fig. 7. Control surface of optimized FLC ($\theta = 0.09$, $\dot{\theta} = 0.12$).

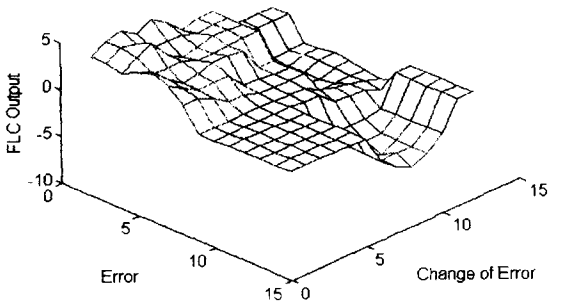


Fig. 8. Control surface of optimized FLC ($\theta = -0.14$, $\dot{\theta} = -0.02$).

with emphasis on some regions which may suggest that it is because of the non-linearity of the inverted pendulum and the different initial conditions.

4. Conclusions

In this paper, we demonstrated the applicability of using genetic algorithms to reduce and optimise rule base controller. We compare the output between the optimised set and the original set of rules. We concluded that the rules are reduced. Meanwhile, the reconstructed controller has better performance than the original one because GAs automatically remove the useless rules and improve the existing ones. As a result, this approach can provide a low-cost and robust means of design of the fuzzy rule-based controller. From the results obtained, the proposed genetic algorithm model has demonstrated its capabilities in terms of high robustness, flexibility and reliability by consistently improving the performance of the fuzzy logic controller.

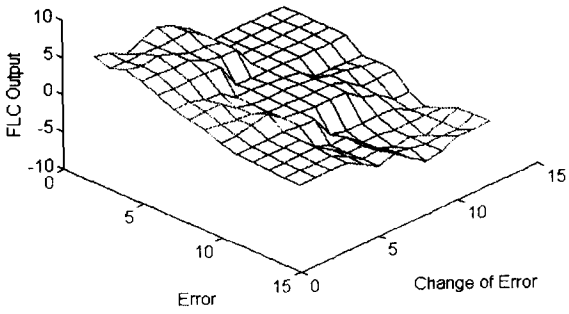


Fig. 9. Control surface of optimized FLC ($\theta = 0.108$, $\dot{\theta} = -0.125$).

References

[1] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley, Reading, MA, 1989).

[2] C. Karr and E. Gentry, Fuzzy control of pH using genetic algorithms, *IEEE Trans. Fuzzy Systems* **1** (1993) 46–53.

[3] M.A. Lee and H. Takagi, Integrating design stages of fuzzy systems using genetic algorithms, *Proc. IEEE Internat. Conf. on Fuzzy Systems* (1993) 612–617.

[4] D. Park, A. Kandel and G. Langholz, Genetic-based new fuzzy reasoning models with application to fuzzy control, *IEEE Trans. Systems Man Cybernet.* **24** (1994) 39–47.

[5] P. Wang and D.P. Kwok, Auto-tuning of classical PID controllers using an advanced genetic algorithm, IECON'92, Vol. 3 (1992) 1224–1229.