J Syst Eng (1991)1:114–118

Journal of Systems Engineering

Optimum Design of Fuzzy Logic Controllers Using Genetic Algorithms

D.T. Pham and D. Karaboga

Intelligent Systems Research Laboratory, School of Electrical, Electronic and Systems Engineering, University of Wales, PO Box 904, Cardiff, CF1 3YH, UK

This paper describes the use of genetic algorithms (GAs) to design fuzzy logic controllers (FLCs). Two GAs are considered, a basic GA and a modified GA. The modified GA employs 'cross breeding' between different gene populations to produce a fitter population. Simulation results are presented which demonstrate the ability of the modified GA to design better FLCs than those obtained with the basic GA.

Keywords: Fuzzy control; Genetic algorithms; Expert control; Optimal controller design

1. Introduction

Fuzzy logic control is based on Zadeh's theory of fuzzy sets [1]. Since fuzzy logic control was first introduced by Mamdani [2], several fuzzy logic controllers (FLCs) have been designed for diverse practical applications (see, for example, [3–5]).

The key element in an FLC is a set of control rules or a relation matrix representing the mapping between the FLC's input fuzzy error values and output fuzzy control actions.

This paper describes the use of genetic algorithms (GAs) to optimise the relation matrix of an FLC. Basic GAs were initially considered by the authors for this task in [6]. In the present work, a modified GA is employed which implements the optimisation in two stages. The first stage produces independent populations of 'genes' representing preliminary FLC designs. A 'cross breeding' operation is performed to generate a population of strong genes for further optimisation in the second stage.

The main body of the paper comprises four sections. Section 2 reviews some of the previous work on the design of FLCs. Section 3 provides a brief introduction to fuzzy control. Section 4 describes both the basic GA used in the previous work and the modified GA. Section 5 presents the results of computer simulations of the responses of a time-delayed second-order plant under the control of FLCs designed by the basic and the modified GAs.

2. Previous Work on FLC Design

Researchers have concentrated on the problem of extracting control rules for FLCs. Manual extraction of rules has two major difficulties. First, experienced operators from whom rules can be acquired may not be readily available. Second, such operators may not be able to represent their process control knowledge as accurate and consistent rules. Thus, efforts have been devoted to finding methods to extract rules automatically. For example, Procyk and Mamdani [7] have described a self-organizing FLC capable of this. Lee and Berenji [8] have reported a self-learning FLC employing reinforcement techniques to learn the required rules. Automatic rule learning has also been achieved in the work by Patrikar and Provence [9] who have used neural networks for this purpose.

The automatic generation of control rules based on a model of the behaviour of the controlled process rather than the operator's actions or experience has also been attempted. Work in this area includes

Received 2 July 1991

Correspondence and offprint requests to: D.T. Pham, Intelligent Systems Research Laboratory, School of Electrical, Electronic and Systems Engineering, University of Wales, PO Box 904, Cardiff, CF1 3YH, UK

that by Czogala and Pedrycz [10] and Chen [11]. The latter author has employed a method known as the cell-state-space method for constructing the rule base. Peng [12] has described a parametric function optimisation method for deriving control rules for systems with known mathematical models. FLCs designed in this way have been shown by the author to possess a better performance than the conventional PID controllers available for such systems.

3. Fuzzy Logic Control

The basic structure of an FLC is conceptually shown in Fig. 1. The knowledge base of the FLC is its relation matrix R which, as previously mentioned, represents the rules for controlling the plant. Using R, the computation unit produces a fuzzy output B' from a fuzzy input A'. B' is defuzzified by the defuzzification unit to give the output v to control the plant. A' is obtained from the fuzzification unit. The input to the latter is the error between the desired plant output r and the actual plant output y.

During operation of the controller, if the fuzzified observed plant error is A', the controller's fuzzy output B' will be produced by the computation unit according to the compositional rule of inference [1] as follows:

$$B' = A' \circ R$$

A' ○ R is the sup-star composition of A' and R defined by using either the Max–Min operator or the Max–Product operator [6].

The Max–Product operator has been adopted in this study as it has been shown to produce smoother control actions [13].

The fuzzification unit converts a 'crisp' error value u ($-5 \leq u \leq +5$) into a fuzzy set A' the elements of which are linguistic variables. The following seven linguistic variables were used in this work: Negative Large (NL) or u_1, Negative Medium (NM) or u_2, Negative Small (NS) or u_3, Zero (ZE) or u_4, Positive Small (PS) or u_5, Positive Medium (PM) or u_6, Positive Large (PL) or u_7. These linguistic variables are themselves fuzzy sets defined by the membership functions shown in Fig. 2. Note that, as A' comprises up to seven elements, the number of rows, m, of R is equal to seven.

The output B' of the computation unit is a fuzzy set. The maximum number of elements of B' is equal to the number of columns, n, of R, which has been chosen as 11 in this work. That is, the output space of the FLC has been quantised into 11 different levels in the interval $[-5, +5]$.

The defuzzification unit converts B' into a crisp value for controlling the plant. The centre-of-area method has been selected to implement the defuzzification [6].

4. Genetic Algorithms

This section outlines the operation of the GA used by the authors in a previous study [6] and presents the improved GA employed in the present work.

4.1. Previous GA

Figure 3 shows the structure of the GA. There are five basic components: a random number generator, a 'fitness' evaluation unit and genetic operators for the 'reproduction', 'crossover' and 'mutation' operations.

The initial population required at the start of the GA is a set of number strings or genes produced by the random number generator. Each string is a one-dimensional representation of a matrix R, of the form

$$\mu_{11}\, \mu_{12} \cdots \mu_{ij} \cdots \mu_{mn}$$

where μ_{ij} is an element of the relation matrix R. μ_{ij} is itself a string of binary digits.

Associated with each string is a fitness value as computed by the evaluation unit. A fitness value is a measure of the goodness of the relation matrix that the string represents. The aim of the genetic

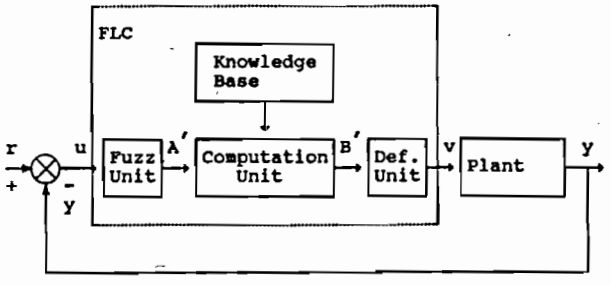

Fig. 1. Basic structure of a fuzzy control system.

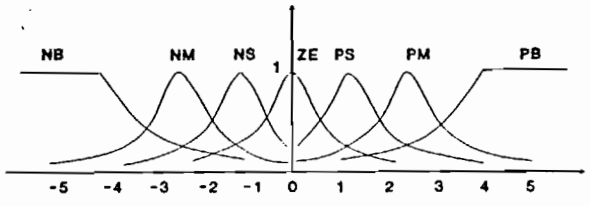

Fig. 2. Membership functions for fuzzy sets of input errors.

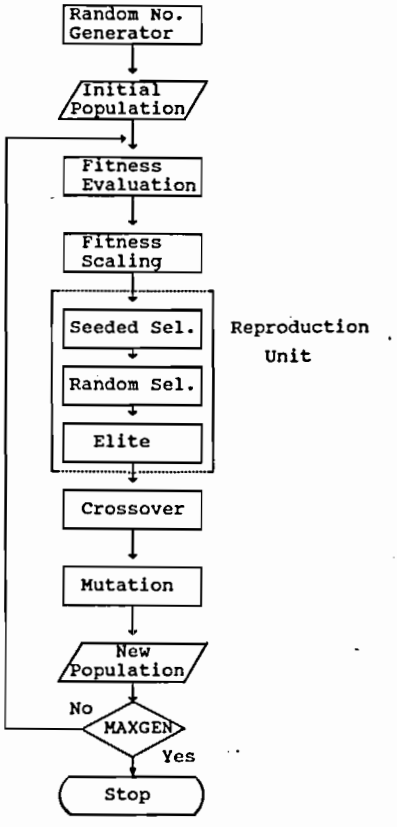

Fig. 3. Flow chart of basic GA.

operators is to transform a set of strings into sets with higher fitness values.

The basic reproduction operator implements a natural selection function known as 'seeded selection'. Individual strings are copied from one set (representing a generation of solutions) to the next according to their fitness values, the higher the fitness value, the greater the probability of a string being selected for the next generation.

The crossover operator chooses pairs of strings at random and produces new pairs. The simplest crossover operation is to cut the original 'parent' strings at a randomly selected point and exchange their tails. This operation is shown in Fig. 4.

The mutation operator, illustrated in Fig. 5, randomly reverses the values of bits in a string.

In addition to the aforementioned basic components, the GA used also incorporates a number of enhanced features. The 'fitness scaling' unit shown in Fig. 3 is for normalising the fitness values computed by the evaluation unit. A 'scaling window' is used in this normalisation process to distinguish between good and better solutions. The reproduction unit also implements a random selection procedure, controlled by a parameter called the 'generation gap', and an 'elite' procedure for preserving the

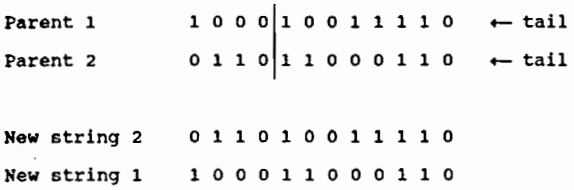

Fig. 4. Simple crossover operation.

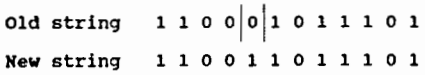

Fig. 5. Mutation operation.

fittest solution in each generation, in addition to the seeded selection process described earlier.

A phase of the GA consists of sequentially applying the evaluation, reproduction, crossover and mutation operations. A new generation of solution strings is produced with each phase of the GA. The GA is repeated for a maximum number of generations, MAXGEN. For more details of GAs, see [14–15].

The GA parameters employed were as follows:

1. Fitness evaluation criterion = ITAE[1]
2. Scaling window = 1
3. Crossover rate = 0.95
4. Mutation rate = 0.01
5. Generation gap = 1.0
6. Length of a solution string = 616 bits[2]
7. Number of solution strings in each generation = 100
8. Maximum number of generations = 400

4.2. Improved GA

The flow chart of the new GA is depicted in Fig. 6. Blocks 1 to n are each identical to the flow chart shown in Fig. 3. These blocks represent n GAs executing in parallel. The initial population for the GAs are created using random number generators with different 'seeds'. After a fixed number of generations, MAXGEN1, the execution of these GAs is stopped. The n populations of solution

[1]The performance of an FLC produced by the GA was measured according to the error in the step response of the plant that it controls. The 'Integral of Time Multiplied by Absolute Error' criterion was used to compute this error [16].

[2]It is recalled that each solution string represents a relation matrix. As mentioned in Section 3, the matrix dimensions are 7×11, giving 77 elements. The string length is the result of assigning eight bits per element.

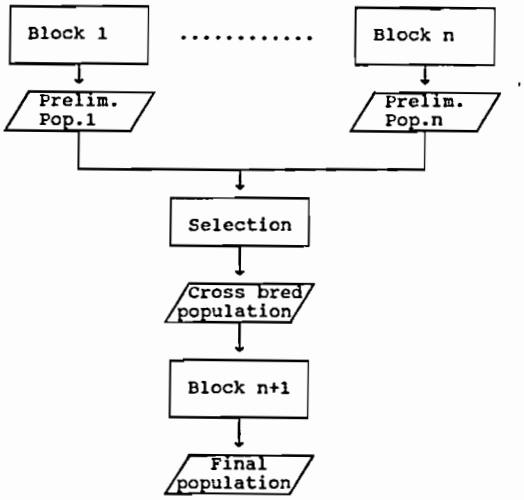

Fig. 6. Flow chart of new GA.

strings produced by these GAs represent n sets of 'preliminary' designs. A procedure is applied to create a new population from these preliminary designs. Various methods could be used for this purpose, including those described in Section 4.1. In this study, for simplicity, the fittest $1/n$ of each population is selected to form the new population. The 'crossbred' population thus generated is used as the initial population for the GA represented by block $(n + 1)$ in Fig. 6. Block $(n + 1)$ is the same as the flow chart in Fig. 3, except that it does not use a random generator to produce the initial population. The GA of block $(n + 1)$ is then executed MAXGEN2 times. The population that it yields is the set of final 'detailed' designs. The fittest solution string is taken as representing the desired R matrix.

In this study,

$$n = 2$$

$$\text{MAXGEN1} = 25$$

$$\text{MAXGEN2} = 350$$

Other GA parameters are as given in Section 4.1.

5. Simulation Results

Figures 7–8 show the simulated step responses of a time-delayed second-order plant with transfer function

$$G(s) = \frac{\exp(-0.4s)}{(0.3s+1)^2}$$

when under the control of two different FLCs designed by the unmodified GA described in Section 4.1. To obtain the relation matrices for these FLCs, the GA was run using two different initial populations. The GA parameters employed, including the maximum number of generations, are as given in Section 4.1.

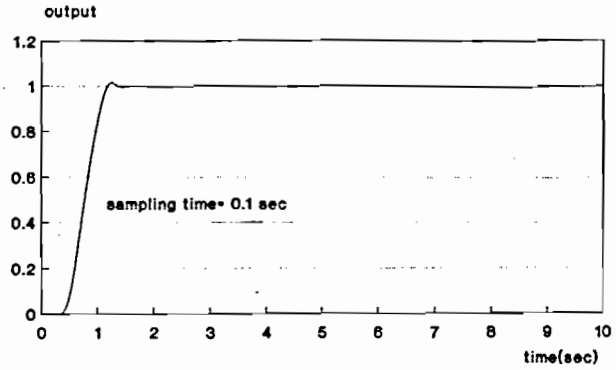

Fig. 7. Step response obtained using first FLC designed by basic GA.

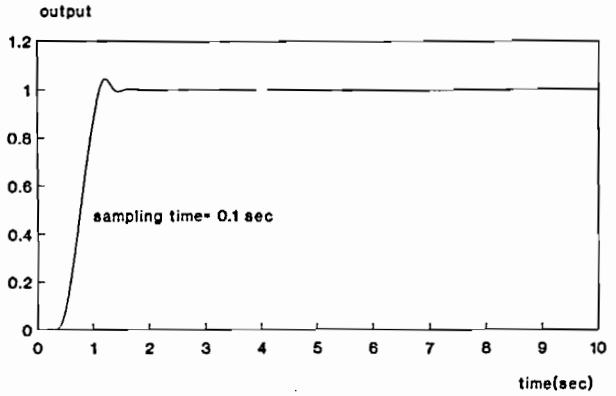

Fig. 8. Step response obtained using second FLC designed by basic GA.

The step response of the same plant under the control of a FLC designed by the new GA is presented in Fig. 9. Note that the new control system is faster than the previous two systems, although the overshoot incurred is higher. This is a result of the ITAE optimisation criterion adopted.

The effectiveness of the new GA is more evident in Fig. 10 which shows the evolution of the fitness ratio r_i defined as

$$r_i = \frac{\text{Min ITAE (overall)}}{\text{Min ITAE } (i\text{th generation})}$$

It can be seen that the two r_i plots for the unmodified GA reach lower saturation levels than that attained by the r_i plot for the new GA.

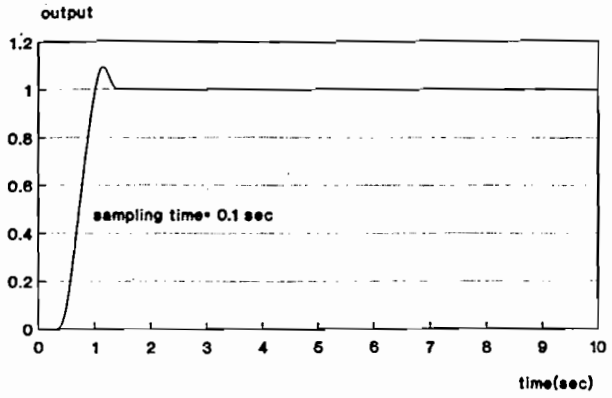

Fig. 9. Step response obtained using FLC designed by new GA.

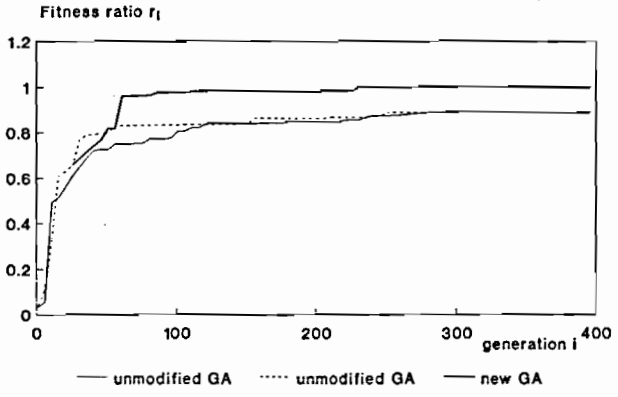

Fig. 10. Evolution of fitness ratio r_i.

6. Conclusion

This paper has focussed on an improved genetic algorithm for designing fuzzy logic controllers. The improvement is based on the idea of dividing the design process into two stages, a preliminary and a detailed design stage, and 'cross breeding' the preliminary design solutions to form a set of good alternatives for 'detailed' consideration. Results obtained for the control of a time-delayed second-order plant have demonstrated the ability of the new GA to find better solutions than could be obtained from the unmodified GA.

Acknowledgement

The authors would like to thank Erciyes University, Turkey, for supporting D. Karaboga.

References

1. Zadeh L. Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans Syst Man Cybern 1973; SMC-3: 28–44
2. Mamdani EH. Applications of fuzzy algorithms for control of simple dynamic plant. Proc IEE 1974; 121(12): 1585–1588
3. Sutton R, Towill DR. An introduction to the use of fuzzy sets in the implementation of control algorithms. J Inst Electronic Radio Eng 1985; 55(10): 357–367
4. Holmblad LP, Ostergaard LL. Control of a cement kiln by fuzzy logic. In: Gupta MM, Sanchez E (eds) Fuzzy information and decision process, North-Holland, Amsterdam, 1982
5. Huang LJ, Tomizuka M. A self-paced fuzzy tracking controller for two-dimensional motion control. IEEE Trans Syst Man Cybern 1990; 20(5): 1115–1124
6. Pham DT, Karaboga D. A new method to obtain the relation matrix of fuzzy logic controllers In: Proc. 6th Int. Conf. on Applications of Artificial Intelligence in Engineering (AIENG91), Oxford, UK, July 1991, pp 567–581
7. Procyk TJ, Mamdani EH. A self-organizing linguistic process controller. Automatica 1979; 15: 15–30
8. Lee CC, Berenji HR. An intelligent controller based on approximate reasoning and reinforcement learning, Proc IEEE Int symp on intelligent control, Albany, NY, 1989, pp 200–205
9. Patrikar A, Provence J. Neural network implementation of linguistic controllers, Proc. 12th IASTED Int. Symp. on robotics and manufacturing, Santa Barbara, CA, 1989
10. Czogala E, Pedrycz W. Fuzzy rules generation for fuzzy control. Cybern Syst 1982; 13: 275–293
11. Chen YY. Rules extraction for fuzzy control systems. In: IEEE Int. Conf. on Systems Man and Cybernetics 1989; vol 2, pp 526–527
12. Peng XT. Generating rules for fuzzy logic controllers by functions. Fuzzy Sets Syst 1990; 36: 85–89
13. Yamazaki T. An improved algorithm for a self organising controller, PhD thesis, Queen Mary College, University of London, 1982
14. Holland JH. Adaptation in natural and artificial systems, The University of Michigan Press, Ann Arbor, MI, 1975
15. Goldberg DE. Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, Reading, MA, 1989
16. Ogata K. Modern control engineering, Prentice-Hall, Englewood Cliffs, NJ, 1970