On Simplifying the Automatic Design of a Fuzzy Logic Controller

France Cheong
School of Business IT,
RMIT University, Melbourne, Victoria 3000, Australia.
email : france.cheong@rmit.edu.au

Richard Lai
Department of Computer Science and Computer Engineering,
La Trobe University, Bundoora, Victoria 3083, Australia
email : lai@cs.latrobe.edu.au

Abstract

With the availability of a wide range of Evolutionary Algorithms such as Genetic Algorithms, Evolutionary Programming, Evolution Strategies and Differential Evolution, every conceivable aspect of the design of a fuzzy logic controller has been optimized and automated. Although there is no doubt that these automated techniques can produce an optimal fuzzy logic controller, the structure of such a controller is often obscure and in many cases these optimizations are simply not needed. We believe that the automatic design of a fuzzy logic controller can be simplified by using a generic rule base such as the MacVicar-Whelan rule base and using an evolutionary algorithm to optimize only the membership functions of the fuzzy sets. Furthermore, by restricting the overlapping of fuzzy sets, using triangular membership functions and singletons, and reducing the number of parameters to represent the membership functions, the design can be further simplified. This paper describes this method of simplifying the design and some experiments performed to ascertain its validity.

Keywords Evolutionary Algorithms, Evolutionary Programming, Fuzzy Logic Controller, Fuzzy Systems

1. Introduction

Fuzzy logic controllers (FLCs) are rule based systems that use fuzzy linguistic variables (e.g. *small*, *large*, etc) to model human rule-of-thumb approaches to problem solving. They have been successfully applied to many control problems because no mathematical modelling is involved; only heuristic knowledge is required. The main problem with FLCs is that there is no generalised design method; their design has been an ad hoc trial and error exercise for a long time. With the advent of global optimization techniques such as Genetic Algorithms (and other evolutionary algorithms) many parts of the design have been optimized and automated such as the derivation of the rule base, the minimization of the number of rules and the membership functions, etc. Every conceivable aspect of the design process has been the object of optimization. Although there is no doubt that these optimization techniques can produce optimum FLC designs, often these designs cannot be interpreted meaningfully.

We believe that FLC design has been the subject of too much automation; in particular, we feel that in many cases it is not necesssary to optimize the design of the rule base provided that a sound template rule base such as the MacVicar-Whelan rule base [1] is used. This rule base is a standard template rule base built according to common engineering sense and experience with fuzzy logic. It defines a reasonable set of rules that can be adjusted by excluding, modifying or adding new control rules based on the specificity of the control problem. If the input variables to the FLC are the error and change in error, then the rule base can be built from the following MacVicar-Whelan meta rules:

1. If both the error and change in error are zero, then change in output is zero.
2. If the error is tending to zero at a satisfactory rate, then change in output is zero.
3. If the error is not self-correcting, then change in output is not zero and depends on the sign and magnitude of the error and change in error.

0-7803-7461-4/02/$17.00 © 2002 IEEE

The rule base can be formulated using these meta rules and a 7 by 7 rule base expressed as a fuzzy associative matrix (FAM) [2] is shown in Figure 1 (see [1, 3] for a detailed formulation of the rule base).

Error \ Change in Error	NB	NM	NS	ZE	PS	PM	PB
NB	1	1	1	1	2	3	4
NM	1	2	2	2	3	4	5
NS	1	2	3	3	4	5	6
ZE	1	2	3	4	5	6	7
PS	2	3	4	5	5	6	7
PM	3	4	5	6	6	6	7
PB	4	5	6	7	7	7	7

Figure 1. 7 by 7 MacVicar-Whelan rule base

In this paper, we aim to show that the automatic design of FLCs for most control applications can be simplified by using a fixed rule base such as the MacVicar-Whelan rule base and adjusting only the membership functions of the fuzzy sets using an Evolution Algorithm such as Evolutionary Programming. We believe that the adjustment of the membership functions can provide enough latitude to meet the requirements of the control problem. Should the adjustment of the membership functions not give satisfactory results, then optimization of the rule base might be contemplated. However, it is believed that this will not often be the case.

We demonstrate the validity of our simplified approach by designing FLCs to control three plant processes with second order transfer functions that we used in a previous study. Then we assessed the performance of the FLCs by simulating step responses and compared them with those obtained in a previous study where the membership functions and the rule base were both optimized.

2. Evolutionary Algorithms and Evolutionary Programming

Evolutionary Algorithms (EAs) are a class of algorithms that use some of the known mechanics of evolution, more specifically the processes of selection, reproduction and mutation to search for the best solution to a problem. The interest in EAs is mainly due their flexibility, adaptibility and robustness in solving difficult optimization problems. Unlike many classical optimizing techniques, EAs do not require the computing of local derivatives to guide the search process; only an objective function needs to be computed. Furthermore, EAs are more likely to arrive at the global optimum because they work on a population of points instead of a point by point approach as used by conventional optimization techniques.

A typical EA is shown in Figure 2. Given an optimization problem, the parameters concerned are grouped into a structure (an individual) and a collection of such individuals (a population) is created by either randomly generating the parameters or using expert knowledge about the problem. The EA runs iteratively on the population of individuals using the genetic operators in a random way but based on the fitness of the structures to perform such tasks as selecting, copying, exchanging and perturbing portions of individuals to create new generations of individuals and eventually find the best individual representing the solution to the problem. Currently, there are three main EAs: Genetic Algorithms (GAs), Evolution Strategies (ESs) and Evolutionary Programming (EP). They differ in the representation of the problem and the use of genetic operations (selection, reproduction and mutation).

EP is an EA that was conceived by Fogel et al. [4]. EP has traditionally used representations that are tailored to the problem domain such as real-valued vectors for real-valued optimization problems, ordered lists for travelling salesman problems and graphs for finite state machine problems. EP is an abstraction of the evolution process at the species level and not at the individual level like GAs and ESs. Thus, it does not use the recombination mechanism at all since recombination does not occur within species.

An offspring is created by mutating a parent and this can be expressed mathematically as:

$$\sigma^{t+1} = \sqrt{\alpha + \beta * fitness(X^t)} \quad \text{and} \qquad (1)$$

$$X^{t+1} = X^t + N(0, \sigma^{t+1}) \qquad (2)$$

where α and β are system parameters. Since it is sometimes hard to set these parameters, there has been several studies [5, 6, 7] on the use of adaptive system parameters to optimize them. When adaptive system parameters are used, the system parameters are evolved together with the parameters of the

```
Algorithm EA
1.  Start with a randomly initialized population
2.  Evaluate fitness of individuals
3.  While not done do
        3.1 Increment generation counter
        3.2 Select parents for reproduction
        3.3 Recombine genes of selected parents
        3.4 Mutate population stochastically
        3.5 Evaluate fitness of individuals
        3.6 Select survivors
    End while
```

Figure 2. Generic Evolutionary Algorithm

problem to be solved.

Selection of survivors is done by stochastic methods via a tournament based on fitness. If there are N individuals in the population, each individual is mutated to produce an offspring. Each individual from the $2N$ (parents plus offsprings) population is entered into a competition against a pre-selected number of opponents and receives a "win" if its fitness is equal to or better than its opponent. N individuals with the highest "wins" are then selected as the survivors for the next generation. More details on EP can be found in [4, 8, 9].

3. The Simplified Design

The simplified design involves using a fixed MacVicar-Whelan rule base and optimizing the membership functions of fuzzy sets using EP. We chose EP for optimizing the membership functions because EP is designed to work with real numbers as opposed to GAs which traditionally operate on binary representations.

To further simplify the design of FLCs, we use triangular membership functions for the inputs and singletons [10] for the outputs. We also use fuzzy sets with a degree of overlapping of two as shown in Figure 3. This results in the use of only one parameter to describe a fuzzy set (the two other parameters required for triangular fuzzy sets are defined by the neighbouring fuzzy sets). Furthermore, since we use a universe of discourse normalized to the range [-1.0, 1.0] and we avoid the use of trapezoidal fuzzy sets for the first and last fuzzy set, we fix the apices of the first and last fuzzy sets to -1.0 and 1.0 respectively. This further reduces the number of parameters to represent the membership functions by two.

Our simplified design method results into an appreciable reduction of parameters to be optimized as compared to other methods reported in the literature. For example, only 15 (3 x 5) real-valued parameters have to be optimized for an FLC with two inputs and one output and with seven fuzzy sets per input/output variable. In several design methods reported in the literature (e.g. [11, 12]), the use of three parameters per triangular fuzzy set requires the optimization of 63 (3 x 21) real-valued parameters for the membership functions, and if rule base optimization is required, an additional 49 (7 x 7) integer-valued parameters is required such that in total 112 parameters have to be optimized. There is no doubt that it is easier to optimize 15 parameters instead of 112.

Constraining the degree of overlapping of the fuzzy sets to two not only simplifies the representation of the membership functions, it also makes the number of rules firing at one time independent of the number of fuzzy sets per input variable. In a two-inputs FLC, this limits the number of fired rules to a maximum of four irrespective of the number of fuzzy sets used per input variable. In an unconstrained situation, increasing the number of fuzzy sets per input variable increases the number of rules firing at one time because each FLC input would be fuzzified into an increasing number of fuzzy sets and this number is dependent on the number of fuzzy sets overlapping each other.

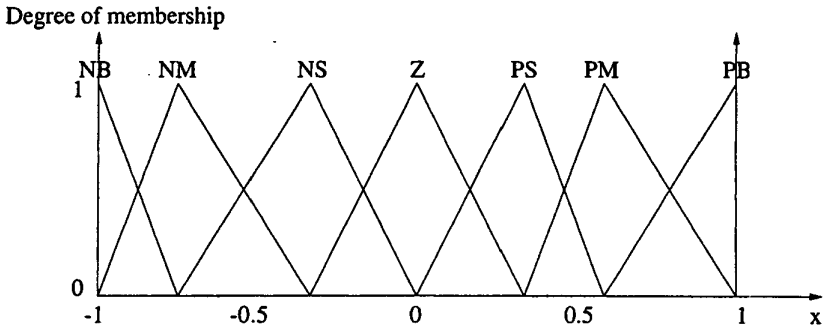

Figure 3. Fuzzy sets with a degree of overlapping of two

This is probably the main reason that has prompted research about reducing the total number of rules in the rule base (e.g. [11, 13, 14, 15, 16]) in an attempt to reduce computation time. However, we believe that constraining the overlapping of fuzzy sets is a more effective method of controlling computation time than a reduced rule base because in an unconstrained situation, the degree of overlapping is not constant across the universe of discourse. Although, reducing the total number of rules in the rule base has a bearing on computation time, it does not affect computation time directly and consistently as when the overlapping of fuzzy sets is constrained. Furthermore, it is safer to keep all the control rules in the rule base rather than reducing them because a full rule base specifies a control action for every possible combination of the inputs.

4. Testing the Design

In order to test the feasibility of our simplified FLC design method, we designed FLCs for controlling unknown plant processes and compared the performance of these FLCs with those designed in a previous study where both the rule base and the membership functions were optimized by an EA. To be able to make some generalizations and comparisons we used the same three plant processes with second order transfer functions used in a previous study. The three plant processes are labelled Plant A, Plant B and Plant C and are defined as follows [1] : $G_A(s) = \frac{2}{s(s+1.4)+2}$, $G_B(s) = \frac{2}{(s+1)(s+2)}$, $G_C(s) = \frac{1}{s(1+0.1s)}$.

[1]The plant transfer functions are unknown to the FLCs which treat the controlled plants as black boxes; they are only used for simulating the unit step responses.

The successful use of EAs requires the choice of the appropriate problem representation, objective function, selection mechanisms, genetic operators and system parameters. Some EAs are fairly straightforward to configure since their operating mechanisms are fixed and only a small number of parameters have to be set. However, others require the selection of mechanisms from a wide available range and have a large number of parameters to be set. EP was chosen to optimize the design of the FLCs because of its ease of configuration and its ability to work directly on the real-valued representation of the problem domain. To keep the experiments simple, we used the standard plain original version of EP, no strategy was used for self-adapting the system parameters.

The problem to be solved by EP is the finding of the membership functions of the three variables *error*, *change in error* and *change in output* of an FLC whose rule base has been determined according to the MacVicar-Whelan meta-rules such that the FLC controls an unknown plant process optimally according to some performance measure. The performance of the FLCs is assessed by means of a unit step response and we used the Integral-of-Time multiplied Absolute-Error (ITAE) [2] as the performance measure.

We used seven fuzzy sets per input/output variable of the FLCs in order to compare the FLCs with the best ones obtained in our previous study. As

[2]ITAE is defined as $\int t\,|e(t)|\,dt$.

discussed above, only one parameter is required to define one fuzzy set and furthermore, since we restrict the parameters of the first and last fuzzy sets to -1.0 and 1.0, only 5 parameters are required per input variable. Thus, a total of 15 parameters is required to define the membership functions of the two inputs and one output of the FLC. Each potential solution to the problem (individual) is represented as a set of 15 real-valued parameters. For an individual to be valid, the parameter set should consist of three subsets of 5 parameters sorted in ascending order, in the range [-1,1] and all values within the subsets should be unique (otherwise there will be less than seven fuzzy sets). We used a population of 100 individuals and initialized the parameters of each individual with random numbers in the range [-1.0, 1.0]. To ensure the validity of the individuals, we grouped their parameters into three subsets (each one representing the membership function of an FLC variable) and sorted them in ascending order.

At the beginning of each EA cycle, individuals are selected to be parents for creating offsprings; in EP all individuals are selected to be parents. At the end of each EA cycle, another selection mechanism is required to select survivors from the population of parents and offsprings to form the next generation. In EP, survivors are selected using a probabilistic function (tournament) based on fitness (see section 2.). We used a tournament size of 10.

EP uses mutation only, it does not use recombination. We used the standard mutation operator as described in section 2. and experimented with several values of α and β and found good values to be 0.0 and 0.1 respectively.

Each solution FLC for the three plant processes was obtained by evolving EP for 100 runs and each run consisted of 300 generations. The performance of the FLCs was judged by comparing the steps responses of FLCs with and without optimized rule bases. Step responses for plants A, B and C are shown in Figures 4, 5 and 6 respectively. It can be seen that the performance of the FLCs designed by our simplified method is equal to that of the FLCs with both optimized rule bases and membership functions except for plant C. Slightly better control of plant C is achieved with an FLC using an optimized rule base. This tends to suggest that in some cases, the generic MacVicar-Whelan rule base may not perform adequately and in these situations rule base optimization might be required.

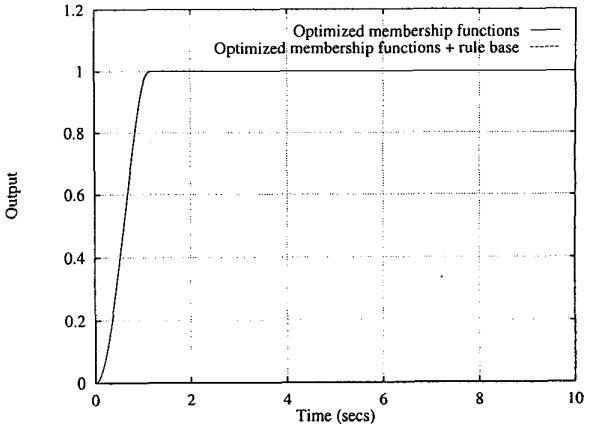

Figure 4. Step Responses for Plant A

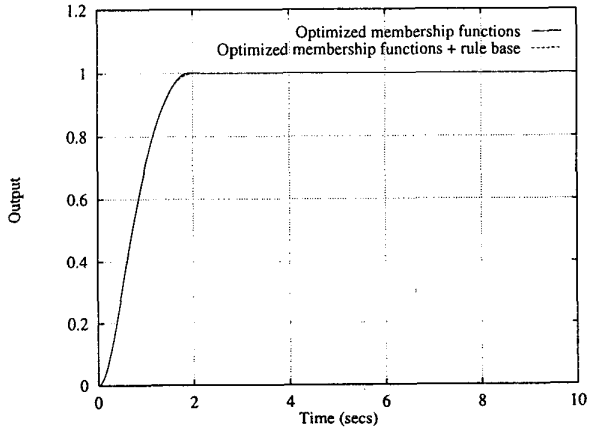

Figure 5. Step Responses for Plant B

5. Conclusions

In this paper, we have presented a method for simplifying the automatic design of an FLC by using a generic MacVicar-Whelan rule base and optimizing only the membership functions of the fuzzy sets with the use of Evolution Programming. The design was further simplified by restricting the overlapping of fuzzy sets, using triangular membership functions and singletons and reducing the number of parameters to represent the membership functions.

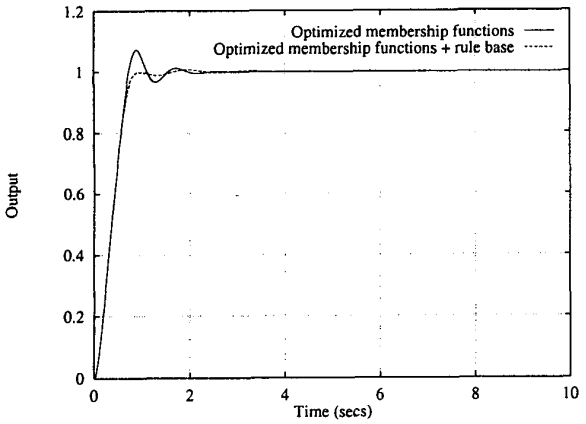

Figure 6. Step Responses for Plant C

Our design method results in the optimization of a highly reduced number of parameters by Evolution Programming. For an FLC with two inputs and one output and with seven fuzzy sets per input/output we need to optimize only 15 parameters as compared to 112 parameters used by other methods. The method not only facilitates the optimization process by using a reduced parameter set; it also results in the design of an FLC with several advantages. First, the rules used by an FLC are clearly understood since we use a generic rule MacVicar-Whelan rule base; second, the number of rules firing at one time is independent of the number of fuzzy sets per input variable because the degree of overlapping of the fuzzy sets has been fixed to two. This obviates the need for reducing the number of rules in the rule base as advocated by several studies. Output computation time is no longer adversely affected by an increase in the number of fuzzy sets per input variable and furthermore a full rule base is safer to use than a partial rule base.

We have tested the method by designing FLCs to control three plant processes and comparing the performance of the FLCs with those having both optimized rule base and membership functions. Based on the experiments carried out, we can affirm that a generic MacVicar-Whelan rule base can be used for most control applications since the performance of FLCs with generic MacVicar-Whelan rule bases was as good as the performance of FLCs with optimized rule base and membership functions in two cases out of three. Thus, we have demonstrated the validity of our method for simplifying FLC automatic designs.

References

[1] P. J. MacVicar-Whelan, "Fuzzy sets for man-machine interaction," *International Journal of Man-Machine Studies*, vol. 8, pp. 687–697, 1976.

[2] B. Kosko, *Neural Networks and Fuzzy Systems.* Englewood Cliffs, NJ: Prentice Hall, 1992.

[3] R. R. Yager and D. P. Filev, *Essentials of fuzzy modeling and control.* New York, NY: John Wiley and Sons, Inc, 1994.

[4] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution.* New York: Wiley Publishing, 1966.

[5] D. B. Fogel, L. J. Fogel, and J. W. Atmar, "Meta-Evolutionary Programming," in *Proc. 25th Asilomar Conference on Signals, Systems & Computers* (R. R. Chen, ed.), (Pacific Grove, CA), pp. 540–545, 1991.

[6] D. B. Fogel, L. J. Fogel, J. W. Atmar, and G. B. Fogel, "Hierarchic methods of Evolutionary Programming," in *Proc. 1st Annual Conference on Evolutionary Programming*, (San Diego, CA), pp. 175–182, Evolutionary Programming Society, 1992.

[7] N. Saravanan, D. B. Fogel, and K. M. Nelson, "A Comparison of Methods for Self-Adaptation in Evolutionary Algirithms," *BioSystems*, vol. 36, pp. 157–166, 1995.

[8] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* Piscataway, NJ: IEEE Press, 1995.

[9] T. Bäck, *Evolutionary Algorithms in Theory and Practice.* New York: Oxford University Press, 1996.

[10] D. I. Brubaker, "Fuzzy-logic basics: intuitive rules replace complex math," *EDNASIA Magazine*, vol. 37, no. 13, pp. 111–116, 1992.

[11] M. A. Lee and H. Takagi, "Integrating Design Stages of Fuzzy Systems using Genetic Algorithms," in *Proc. 2nd IEEE International Conference on Fuzzy systems*, pp. 612–617, 1993.

[12] H. R. Berenji and P. Kheddar, "Learning and Tuning Fuzzy Logic Controllers Through Reinforcements," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 724–740, September 1994.

[13] D. S. Feldman, "Fuzzy Logic Synthesis with Genetic Algorithms," in *Proc. 5th International Conference on Genetic Algorithms*, pp. 312–17, 1993.

[14] H. Ishibuchi and T. Murata, "Minimizing the Fuzzy Rule Base and Maximizing Its Performance by a Multi-Objective Genetic Algorithms," in *Proc. 6th IEEE International Conference on Fuzzy systems*, pp. 259–264, 1997.

[15] R. G. Riccardo Rovatti and G. Baccarani, "An Enhanced Two-Level Boolean Synthesis Methodology for Fuzzy Rules Minimization," *IEEE Transactions on Fuzzy Systems*, vol. 3, pp. 288–299, August 1995.

[16] P. Thrift, "Fuzzy Logic Synthesis with Genetic Algorithms," in *Proc. 4th International Conference on Genetic Algorithms*, pp. 509–513, 1992.