Designing a Hierarchical Fuzzy Logic Controller Using the Differential Evolution Approach

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

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

Abstract

In conventional fuzzy logic controllers, the computational complexity increases with the dimensions of the system variables; the number of rules increases exponentially as the number of system variables increases. Hierarchical fuzzy logic controllers (HFLC) have been introduced to reduce the number of rules to a linear function of system variables. However, the use of hierarchical fuzzy logic controllers raises new issues in the automatic design of controllers, namely the coordination of outputs of sub-controllers at lower levels of the hierarchy. In this paper, a method is described for the automatic design of an HFLC using an Evolutionary Algorithm called Differential Evolution (DE).

The aim in this paper is to develop a sufficiently versatile method that can be applied to the design of any HFLC architecture. The feasibility of the method is demonstrated by developing a two-stage HFLC for controlling a cart-pole with four state variables. The merits of the method are automatic generation of the HFLC and simplicity as the number of parameters used for encoding the problem are greatly reduced as compared to conventional methods.

1 Introduction

Fuzzy logic controllers (FLCs) are rule-based expert systems that use human-like intelligence. They can be used for the control of complex and/or ill-defined systems as they can easily capture the approximate and qualitative aspects of human knowledge and reasoning. The key components of an FLC are the membership functions for defining the fuzzy sets and the rule base for producing the output. Traditionally, these components have been designed using expert heuristic knowledge and trial and error. Since this is a tedious and non-optimal method, many automated techniques have been successfully proposed since.

In conventional FLCs, the number of rules increases exponentially as the number of system variables increases [1]. Hierarchical FLCs (HFLCs) can be used to reduce the number of rules to a linear function of system variables. It consists of dividing a global task into sub-tasks, designing an independent FLC for each sub-task, and, devising a strategy for coordinating the sub-controllers to achieve the global objective. The concept of hierarchy in fuzzy logic control has also been introduced for reasons other than the reduction of number of rules. In certain circumstances, the Mamdani FLC yields unsatisfactory results [2]. To fix this problem, Yager [3] proposed a hierarchy called Hierarchical Prioritized Structure (HPS) where specific fuzzy rules override more general ones. Other reasons for using HFLCs for controlling multivariable systems include scalability of the system and simplicity of design and tuning.

In this paper, a method is described for automating the design of an HFLC using an Evolutionary Algorithm called Differential Evolution. In developing this method, the aim is to make it versatile enough to be applied to any HFLC architecture and not just the one shown in Figure 1a.

The rest of the paper is organized as follows. The literature on hierarchical fuzzy logic controllers is reviewed, followed by an explanation of Differential Evolution. The design method is next explained and its use is then demonstrated by developing a two-stage HFLC for controlling a cart-pole with four state variables.

2 A Review of the Literature on Hierarchical Fuzzy Logic Controllers

The proposed HFLC design approach is based on the HFLC architecture shown in Figure 1a. In this arachitecture, lower level FLCs control closely associated system variables (such as the error and change in error of a particular system variable) while higher level FLCs integrates the outputs of pairs of lower level sub-controllers into an intermediate or final control output.

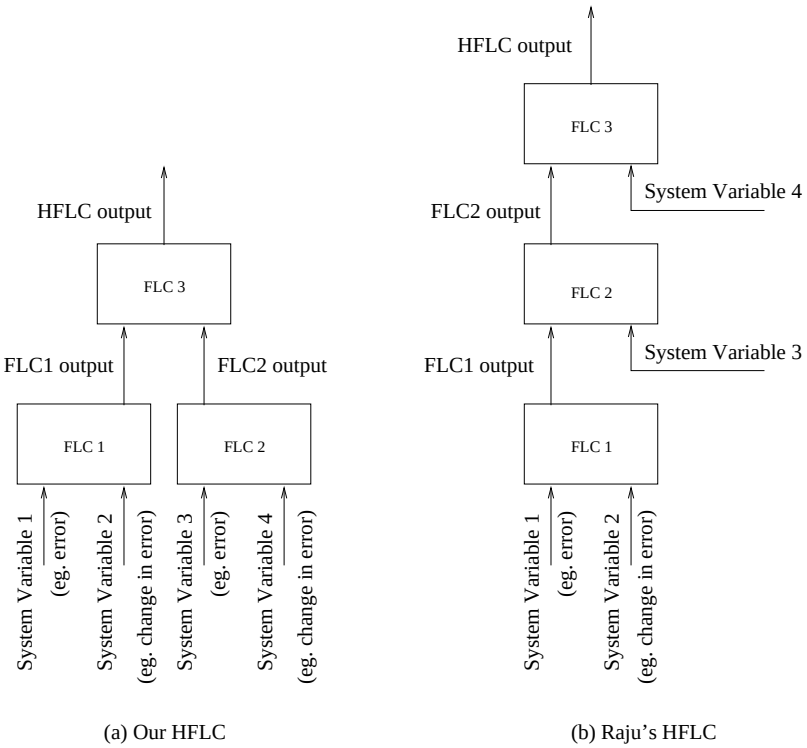


Figure 1: Hierarchical fuzzy logic controller

Raju et al. [1] proposed a hierarchical structure in which the most influential system variables are at the first level, the next most important variables at the next level, and so on. The first level controller gives an approximate output which is modified by the second level controller; the inputs to the latter being the output of the first level controller and other system variables. This process is repeated at succeeding levels of the hierarchy to eventually produce a control output. Our work is different from Raju's work because they used a different HFLC architecture as shown in Figure 1b and furthermore no automation of the design was involved.

Lin et al. [4] automated the design of Raju's HFLC to control the water level in an advanced boiling water reactor. The HFLC was optimized by using the steepest descent method to tune the scaling factors and a method similar to the fuzzy model reference learning control technique [5] to modify the first-level control rules.

Berenji et al. [6] proposed a hierarchical approach that focuses attention on a particular goal at each time instance and which can achieve interacting goals simultaneously. Their "conjunctive goal achievement" technique is in essence similar to Raju's work and no automation of the design was involved as well. Two rule bases were formulated for controlling the cart pole; the first rule base controls the pole angle and the second rule base is activated only if the pole is "approximately" balanced.

Hammell et al. [7] developed a hierarchical architecture for adaptation purposes using Wang and Mendel's method [8] for generating rules. The knowledge base of the model consists of two rule bases: one rule base provides a rough approximation of the system while the other one refines the model. Analysis of system feedback determines when adaptation is necessary and appropriately modifies the rules in either or both of the rule bases.

Horáček et al. [9] extended the Fuzzy Logic-Neural Network (FLNN) proposed by Lin et al. [10] to

the multistage decision case where rules are grouped and groups chained in order to get a hierarchy of decisions. Here, the objective of the hierarchy is to achieve simplification of controller design and tuning. Since decomposition reduces antecedent complexity [11], the designer can fix the most evident rules (or groups of rules) and leave only a few rules to be tuned. Kandadai et al. [12] modified and extended Berenji and Kheddar's GARIC architecture [13] to automatically generate a knowledge base for an HFLC. However, instead of using an FLC for combining the outputs of two lower levels FLCs, they used an exponential function.

Campello et al. [14] presented a hierarchical fuzzy model within the context of an orthonormal basis functions (OBF) framework. OBF is a very promising approach to the areas of non-linear system identification and control, however, OBF fuzzy systems also suffer from the curse of dimensionality problem exhibited by fuzzy systems when applied to large-scale systems. The aim of their work was to bring together the advantages of both hierarchical and OBF frames into a single fuzzy system. They used the Simplified Relational Structure [15], a special kind of fuzzy sub-system which under certain conditions is completely equivalent to a Radial Basis Function (RBF) neural network [16]. The model was used for a predictive control scheme for an ethanol production process and was found to adequately represent the process with a reduced number of free design parameters.

A mutistage fuzzy control system was proposed by Yeh and Li [17] who demonstrated its performance by controlling a cart-pole. Although, the proposed system did reduce the number of rules, the formulation of the rules were not elaborated and the membership functions were not optimized as they were manually designed. Input/output scaling factors were used and were optimized using a Genetic Algorithm.

3 Differential Evolution

Differential Evolution (DE) [18] is a recently developed Evolutionary Algorithm (EA). EAs are characterized by the use of an objective function to guide the search for the best solution to a problem using the mechanics of evolution (such as selection, reproduction and mutation) while DE is characterized by its use of population-derived noise to adapt the mutation rate of the evolution process, simplicity and speed of operation.

In DE, individuals are represented as real-valued vectors. For each generation of the evolution process, each individual (target individual) of the population competes against a new individual (trial individual) for survival to the next generation; only the fitter of the two survives. The trial individual is created by recombining the target individual with another individual created by mutation (mutant individual).

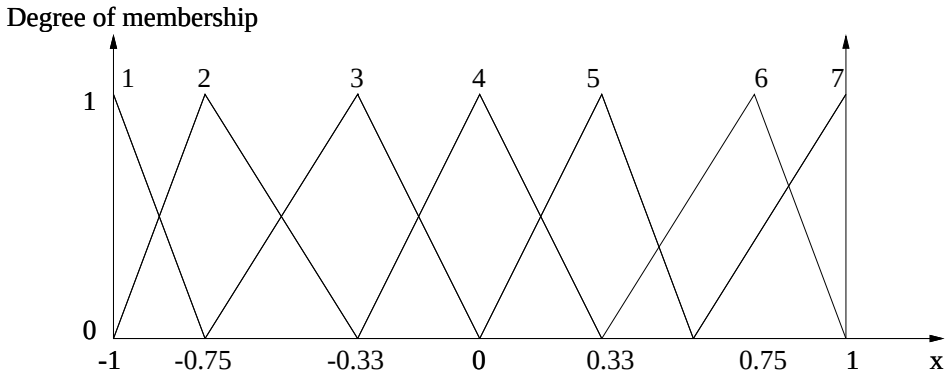
Mutation is performed either on the best individual found so far in the evolution process or any randomly drawn individual from the population. This individual is mutated by adding a perturbation vector to it. The perturbation vector is calculated as the scaled difference between two randomly sampled individuals from the population. Mathematically this can be expressed as: $X_a = X_b + \alpha(X_c - X_d)$, where α is a system parameter.

Recombination creates an offspring (trial individual) by selecting parameters from either the target individual or the mutant individual. Two methods of recombination are used: binary and exponential. In binomial recombination, a series of binomial experiments are conducted to determine which parent contributes which parameter to the offspring. Each experiment is mediated by a crossover constant, CR, where $0 \leq CR \geq 1$. Starting at a randomly selected parameter, the source of each parameter is determined by comparing CR to a uniformly distributed random number from the interval $[0, 1)$. If the random number is greater than CR, the offspring gets its parameter from the target individual, otherwise, the parameter comes from the mutant individual. In exponential recombination, a single contiguous block of parameters of random size and location is copied from the mutant individual to a copy of the target individual to produce an offspring. Starting at a randomly selected parameter, parameters are copied from the mutant individual while a uniformly distributed random number from the interval $[0, 1)$ is less than CR.

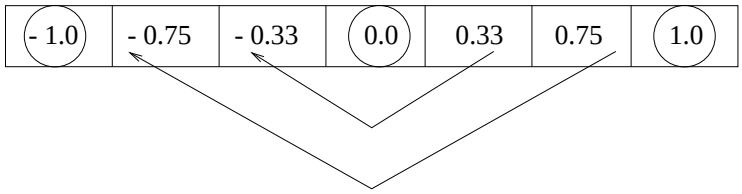
4 The Design Method

Most FLCs used for control purposes must have symmetrical outputs (same magnitude but different signs for inputs of same magnitude and different signs); this fact is often overlooked when FLCs are automatically designed by techniques such as Evolutionary Algorithms and neural networks. In order to produce symmetrical outputs, an FLC must be symmetrical at two levels: the membership functions and the rule base.

To keep the design of FLCs simple, triangular membership functions were used for the inputs and singletons for the outputs. A degree of overlapping of two was also used for the fuzzy sets as shown in Figure 2a. 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 a universe of discourse normalized to the range [-1.0, 1.0] was used and the formation of trapezoidal fuzzy sets were avoided for the first and last fuzzy sets, the apices of the first and last fuzzy sets were fixed to -1.0 and 1.0 respectively.



(a) Symmetrical fuzzy sets with degree of overlapping of 2



(b) Parameters for representing fuzzy sets

Figure 2: Symmetrical membership functions

To achieve symmetricity in the membership functions, the value of the middle fuzzy sets was fixed to 0.0 and an EA was used to generate only the values between the middle and the last fuzzy set. The values for the fuzzy sets between the first and the middle were then replicated from their positive counterparts but with a negative sign as shown in Figure 2b.

Concerning the rule base, as far as possible a template rule base such as the MacVicar-Whelan [19] rule base was used for the sub-systems i.e. the FLCs. This rule base is based on sound engineering principles and has been found to perform satisfactorily for a large variety of control problems. Figure 3 shows a 7 by 7 MacVicar-Whelan rule base.

Should the use of the MacVicar-Whelan rule base prove unsatisfactory, then a new rule base should be designed for the control problem, however, care should be taken to ensure that the designed rule base has symmetrical properties. Figure 4 shows the four steps used in building a symmetrical 7 by 7 rule base. Step 1 consists of generating the rules for the top triangular portion of the rule base. Notice that for the rule base to be symmetrical, the rule at the middle of the matrix should output the middle fuzzy set i.e.

error \ change in error	1	2	3	4	5	6	7
1	1	1	1	1	2	3	4
2	1	2	2	2	3	4	5
3	1	2	3	3	4	5	6
4	1	2	3	4	5	6	7
5	2	3	4	5	5	6	7
6	3	4	5	6	6	6	7
7	4	5	6	7	7	7	7

Figure 3: Standard MacVicar-Whelan rule base for 7 by 7 fuzzy sets

if a 7 by 7 rule base is used, the middle rule should output a fuzzy set encoded as 4 (if the first fuzzy set is numbered as 1). In step 2, all the generated values are replicated to the left of the rule base by transposing rows to columns. Step 3 involves completing the values along the diagonal at the bottom of the matrix. This is achieved by replicating the values already generated along the diagonal. However, the replicated fuzzy sets should be of different "polarity" i.e similar fuzzy sets on the other side of the universe of discourse. In the last step, the remaining cells below the diagonal are filled by transposing and changing the "polarity" of the fuzzy sets generated/replicated in the upper half of the matrix above the diagonal.

The proposed method can be used for building symmetrical FLCs and HFLCs. However, in a previous study [20] it was found out that in many cases FLCs do not require tailor-made rule bases; all that is required is a standard MacVicar-Whelan rule base. A close look at the standard MacVicar-Whelan rule base depicted in Figure 3 shows that it conforms to the notion of a symmetrical rule base. Thus, the proposed method is more likely to be used for designing HFLCs rather than FLCs.

5 Simulation and Results

In this section, the method is applied to the design of an HFLC for controlling the well known cart-pole problem (also known as the inverted pendulum). The cart-pole system shown in Figure 5 consists of a cart moving along a one-dimensional track and a pole attached to the centre of the cart. The pole is free to rotate only in the vertical plane of the cart and track. The objective of the control problem is to apply forces to the cart until it is stationary at the center of the track and the pole is balanced in a vertical position.

The cart pole system is modelled by the following nonlinear differential equations [21, 22]:

$$\ddot{\theta}_t = \frac{gsin\theta_t + cos\theta_t[\frac{-F_t - ml\dot{\theta}_t^2 sin\theta_t + \mu_c sgn(\dot{x}_t)}{m_c+m}] - \frac{\mu_p \dot{\theta}_t}{m_l}}{l[\frac{4}{3} - \frac{mcos^2\theta_t}{m_c+m}]}$$

$$\ddot{x}_t = \frac{F_t + ml[\dot{\theta}_l^2 sin\theta_t - \ddot{\theta}_t cos\theta_t] - \mu_c sgn(\dot{x}_t)}{m_c + m}$$

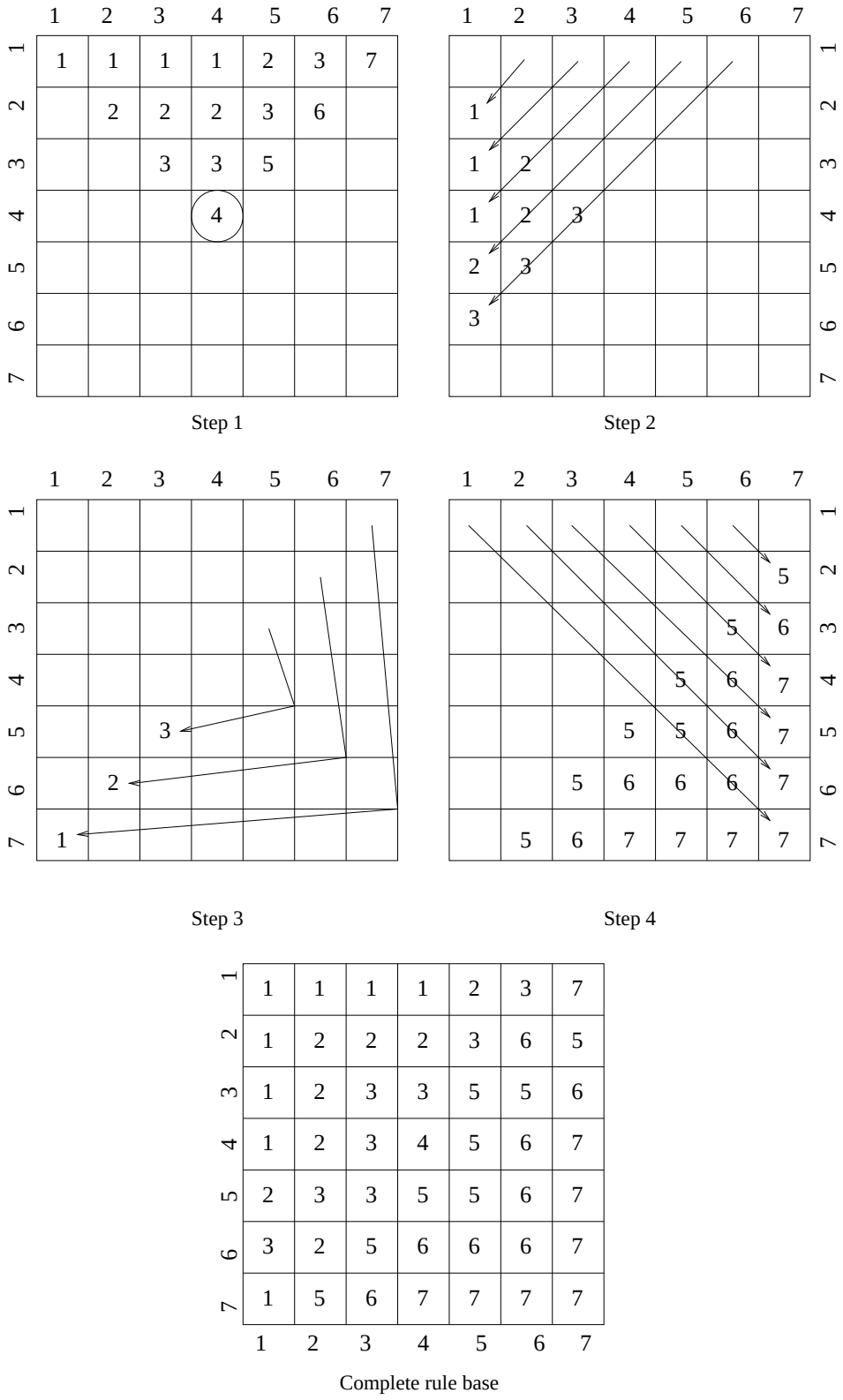


Figure 4: Steps for building a symmetrical rule base

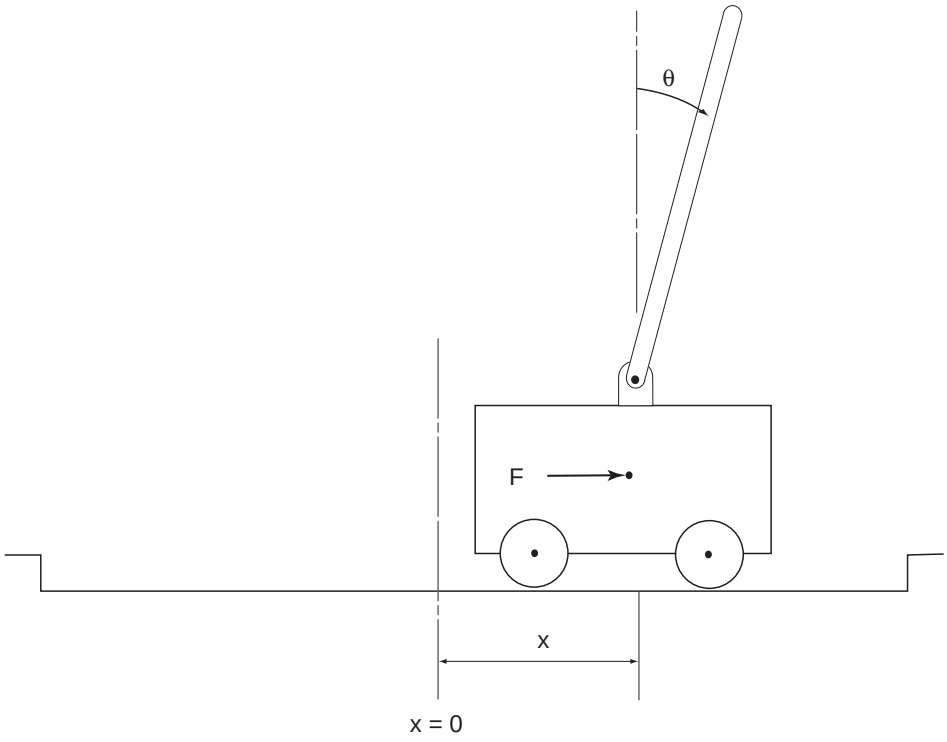


Figure 5: Cart-pole system

where

θ = pole angle;
x = distance of cart from centre of track;
g = -9.8 m/s^2, acceleration due to gravity;
m_c = 1.0 Kg, mass of cart;
m = 0.1 Kg, mass of pole;
l = 0.5 m, half pole length;
μ_c = 0.0005, coefficient of friction of cart on track;
μ_p = 0.000002, coefficient of friction of pole on cart; and
F_t = force ($newtons$) applied to cart's centre of mass at time t.

To build an HFLC for the inverted pendulum, three FLCs must be designed: one FLC for controlling the pole, a second one for controlling the cart and a third one for combining the outputs of the first and second FLCs. The Java programming langauage was used to implement DE and the simulation of the HFLC for controlling the cart-pole. The solution of the differential equations was approximated numerically using Euler's method with a time step of 0.02 second.

As a preliminary step, it was determined whether the MacVicar-Whelan rule base had to be amended for use as independent FLCs for controlling the pole and the cart. The FLCs were trained to control the pole from an initial angle of 15 degrees and the cart from an initial position of 2.4 meters from the center of the track. The pole FLC was tested on the following initial angles: 15, 7.5, 0 , -7.5 and -15 degrees. The cart position controller was tested on the following distances from the centre of the track: 2.4, 1.2, 0, -1.2 and -2.4 meters. Figure 6 shows the performance of the FLCs controlling the pole and the cart separately. It can be seen that the controllers were perfectly symmetrical and furthermore they were able to generalize the control of the pole and the cart from initial positions not used during the optimization process.

Figure 6 clearly shows that the MacVicar-Whelan rule base can satisfactorily control the cart and the pole without any modification. Thus, the problem to be solved consists of optimizing the membership functions of all the input/output variables used by the three FLCs as well as the formulation of a rule base

for combining the outputs of the two lower level FLCs.

7 fuzzy sets were used for each input/output variable and furthermore only used 7 input/output variables were used instead of 9 since the same membership functions were used to describe the outputs of the pole and cart FLCs and the inputs to the higher level FLC. According to the proposed method, only 2 parameters had to be optimized by the EA for a 7-fuzzy-set variable such that in total only 14 parameters have to be optimized. The membership functions were encoded as a vector of 14 real values. The rule base was represented as an array of 7 by 7 integers constrained to contain values in the range [1,7]. For a 7 by 7 rule base, only 15 values had to be optimized, the remaining 34 values were simply derived from the optimized 15 values. Thus, in the method only 29 (14 reals + 15 integers) parameters had to be optimized for an HFLC with 3 FLCs using 7 fuzzy sets per input/output variable. Had 3 parameters per triangular fuzzy set, 7 fuzzy sets per input/output variable, 9 input/output variables and three 7 by 7 rule bases generated from scratch been used, this have required the optimization of 336 (189 reals + 147 integers) values. Furthermore, there is no way such an HFLC can produce symmetrical outputs.

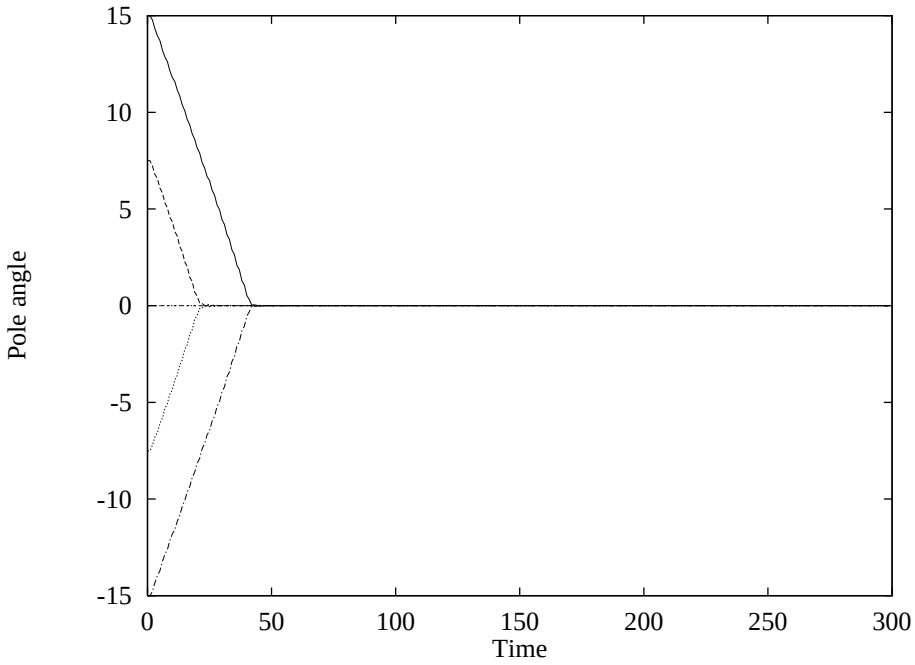
DE was chosen to automate the design of the HFLC because it is simple to use and is very fast. The strategy used for mutation and recombination was based on the best individual found so far (instead of a random individual). Mutation was performed by adding the perturbation vector to the best individual. Recombination consisted of creating a trial individual by conducting a series of binomial experiments to determine whether to inherit parameters from the target or mutant individual.

A population of 50 individuals, $\alpha = 0.7$, and $Cr = 0.7$ were used. The objective function used was the sum of Integral-of-Time-multiplied Absolute-Error (ITAE) of the pole angle and the cart position over 300 time steps. In order to train the HFLC correctly, training situations were needed that would activate the rules in all four quadrants of the rule base, but since the rule base was symmetrical, the number of training situations required were reduced to two quadrants only. The training situations used were: pole angle of 15 degrees, cart position of 2.4 meters and pole angle of -15 degrees, cart position of 2.4 meters. The first training situation used rules in the fourth quadrant while the second one used rules in the second quadrant. The objective score of each training situation was summed to obtain a global objective score. DE performed the optimization within 200 generations.

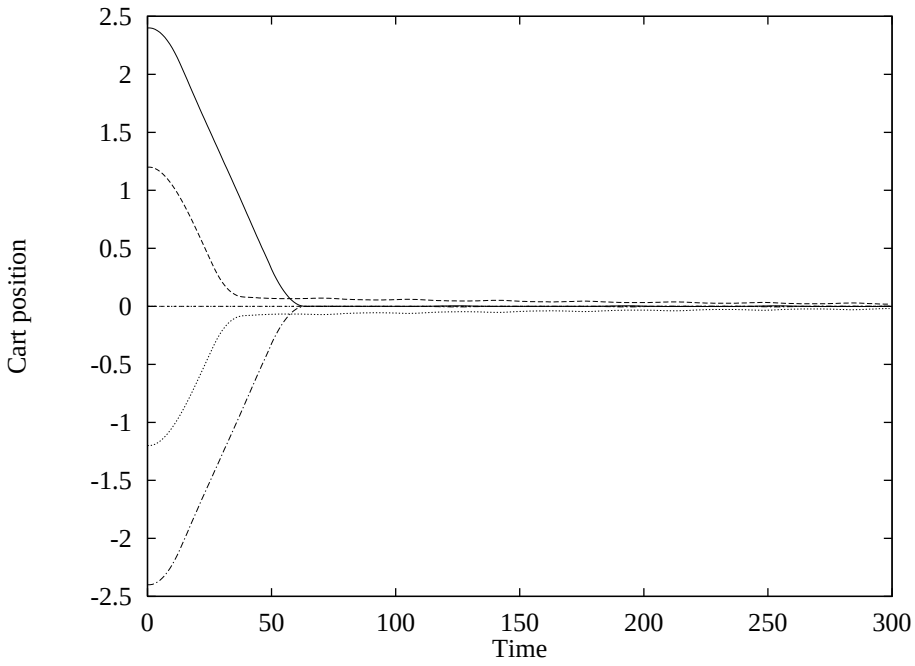
Alternative higher level rule bases for the HFLC were also experimented with and are shown in Figure 7. The use of the standard MacVicar-Whelan rule base was first tried as shown in Figure 7a as the higher level rule base of the HFLC. It was found out that it performed satisfactorily when the pole and the cart required balancing forces in the same direction. This is because it averages the outputs of the two sub-controllers (rules in first and fourth quadrants). However, when the two sub-controllers required opposing forces to keep the pole balanced (such as in cases where the pole was at one extreme position and the cart at the opposite), this averaging process tended to produce a zero balancing force. This was evidenced by the values of 4 (4 is the code for zero output in a set of 7 fuzzy sets) along the diagonal of the rule base.

It was concluded that the rules in the first and fourth quadrants of the MacVicar-Whelan rule base could be used for hierarchical control, and that the rules in the remaining quadrants (or possibly only the rules along the diagonal excluding the centre rule which is fixed) had to be modified. Three different rules bases were experimented with: a manually modified MacVicar-Whelan rule base along the diagonal as shown in Figure 7b, an automatically amended MacVicar-Whelan rule base (only the offending quadrants) as shown in Figure 7c and a whole rule base generated from scratch as shown in Figure 7d according to the method explained in the previous section.

The performance of the HFLCs with the three different rule bases are shown in Figure 8. Each plot shows the performance of the HFLC for five different initial positions of the pole and cart: (15 degrees, 2.4 m), (-15 degrees, -2.4 m), (-15 degrees, 2.4 m), (15 degrees, -2.4 m) and (0 degrees, 0 m). It can be clearly seen that all the plots tend to zero whatever the initial conditions are. Furthermore, the control is perfectly symmetrical and the HFLCs are able to control situations not used during training. Figure 8 also shows that the three rule bases performed equally well. The membership functions found by DE for the three different systems are shown in Tables 1 to 3 and the HFLC control surfaces are shown in Figures 9 to 11.

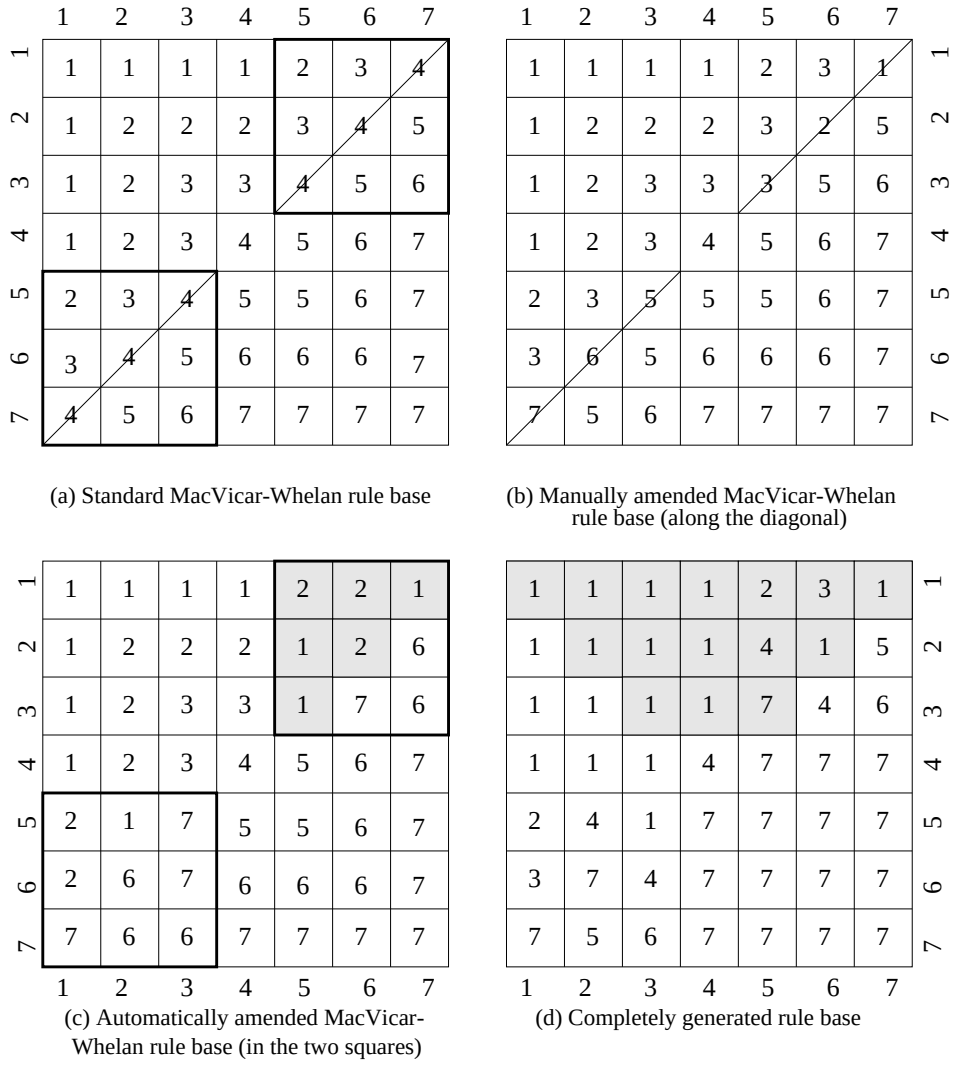


(a) Pole control



(b) Cart control

Figure 6: Independent FLCs



(a) Standard MacVicar-Whelan rule base

(b) Manually amended MacVicar-Whelan rule base (along the diagonal)

(c) Automatically amended MacVicar-Whelan rule base (in the two squares)

(d) Completely generated rule base

Figure 7: Alternative higher level rule bases for HFLCs

Variable	Vertices of fuzzy sets/Singletons						
	1	2	3	4	5	6	7
FLC for controlling pole angle							
Error	-1.0	-0.380823	-0.019173	0.0	0.019173	0.380823	1.0
ChangeInError	-1.0	-0.993443	-0.489002	0.0	0.489002	0.993443	1.0
Output	-1.0	-0.505423	-0.173913	0.0	0.173913	0.505423	1.0
FLC for controlling cart position							
Error	-1.0	-0.990664	-0.989972	0.0	0.989972	0.990664	1.0
ChangeInError	-1.0	-0.438512	-0.381316	0.0	0.381316	0.438512	1.0
Output	-1.0	-0.252943	-0.119085	0.0	0.119085	0.252943	1.0
FLC for combining outputs of angle and cart FLCs							
Angle FLC output	Same parameters as angle FLC output						
Position FLC output	Same parameters as position FLC output						
Combined output	-1.0	-0.185376	-0.077429	0.0	0.077429	0.185376	1.0

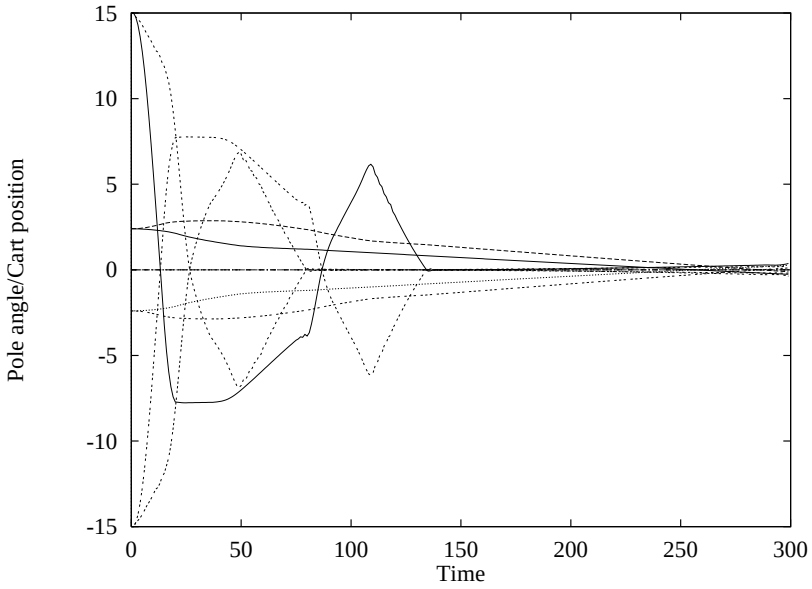
Table 1: HFLC membership functions - Higher level rule base wholly generated

Variable	Vertices of fuzzy sets/Singletons						
	1	2	3	4	5	6	7
FLC for controlling pole angle							
Error	-1.0	-0.143821	-0.053560	0.0	0.053560	0.143821	1.0
ChangeInError	-1.0	-0.982110	-0.969954	0.0	0.969954	0.982110	1.0
Output	-1.0	-0.725724	-0.156310	0.0	0.156310	0.725724	1.0
FLC for controlling cart position							
Error	-1.0	-0.610032	-0.607313	0.0	0.607313	0.610032	1.0
ChangeInError	-1.0	-0.305299	-0.022255	0.0	0.022255	0.305299	1.0
Output	-1.0	-0.807885	-0.804530	0.0	0.804530	0.807885	1.0
FLC for combining outputs of angle and cart FLCs							
Angle FLC output	Same parameters as angle FLC output						
Position FLC output	Same parameters as position FLC output						
Combined output	-1.0	-0.119360	-0.118322	0.0	0.118322	0.119360	1.0

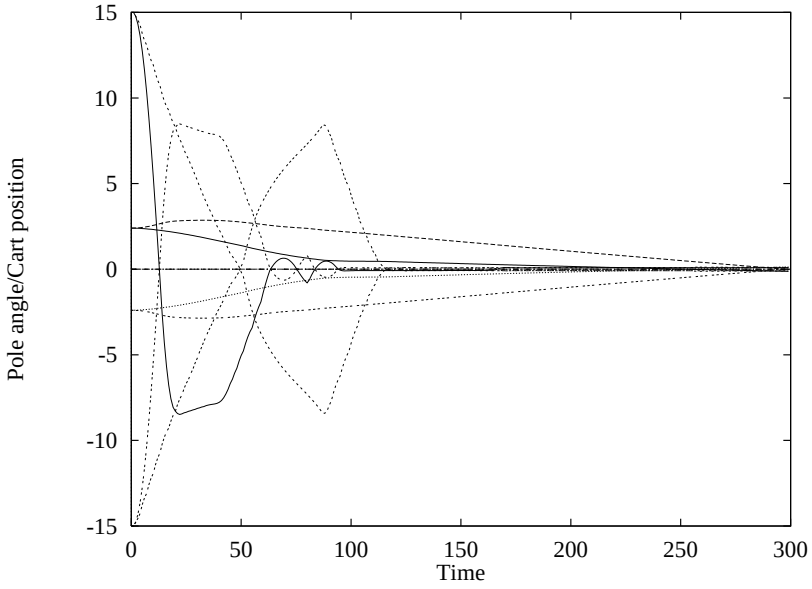
Table 2: HFLC membership functions - Higher level rule base partially generated

Variable	Vertices of fuzzy sets/Singletons						
	1	2	3	4	5	6	7
FLC for controlling pole angle							
Error	-1.0	-0.364545	-0.024755	0.0	0.024755	0.364545	1.0
ChangeInError	-1.0	-0.918990	-0.852036	0.0	0.852036	0.918990	1.0
Output	-1.0	-0.665165	-0.664393	0.0	0.664393	0.665165	1.0
FLC for controlling cart position							
Error	-1.0	-0.997741	-0.994123	0.0	0.994123	0.997741	1.0
ChangeInError	-1.0	-0.295789	-0.194073	0.0	0.194073	0.295789	1.0
Output	-1.0	-0.207256	-0.207241	0.0	0.207241	0.207256	1.0
FLC for combining outputs of angle and cart FLCs							
Angle FLC output	Same parameters as angle FLC output						
Position FLC output	Same parameters as position FLC output						
Combined output	-1.0	-0.339700	-0.303941	0.0	0.303941	0.339700	1.0

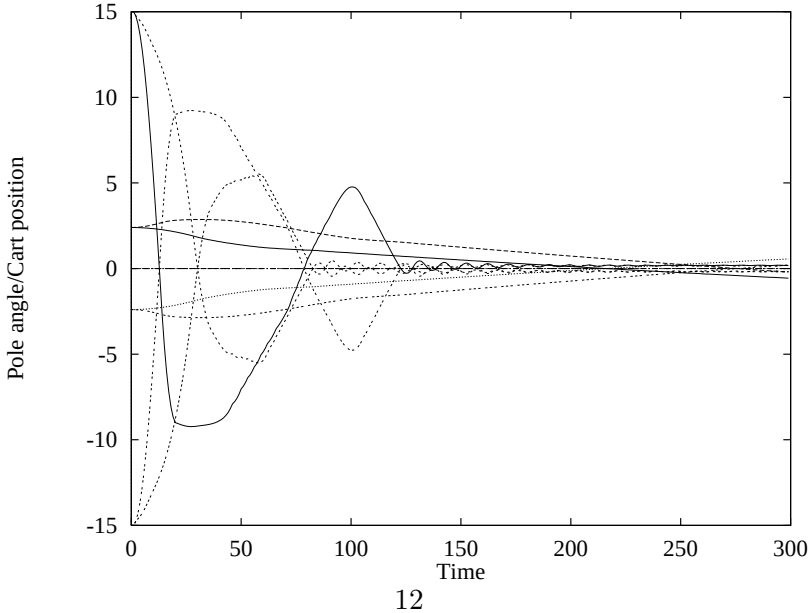
Table 3: HFLC membership functions - Higher level rule base manually designed



(a) Higher level rule base wholly generated

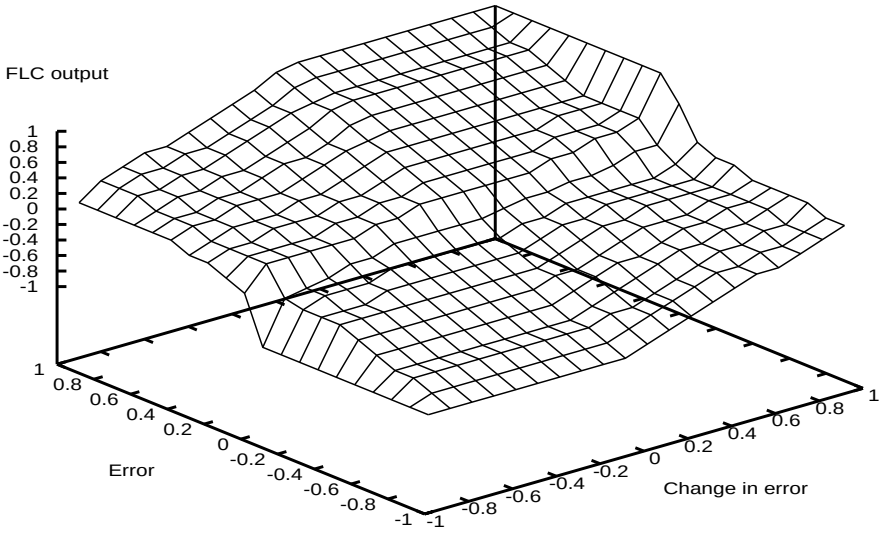


(b) Higher level rule base partially generated

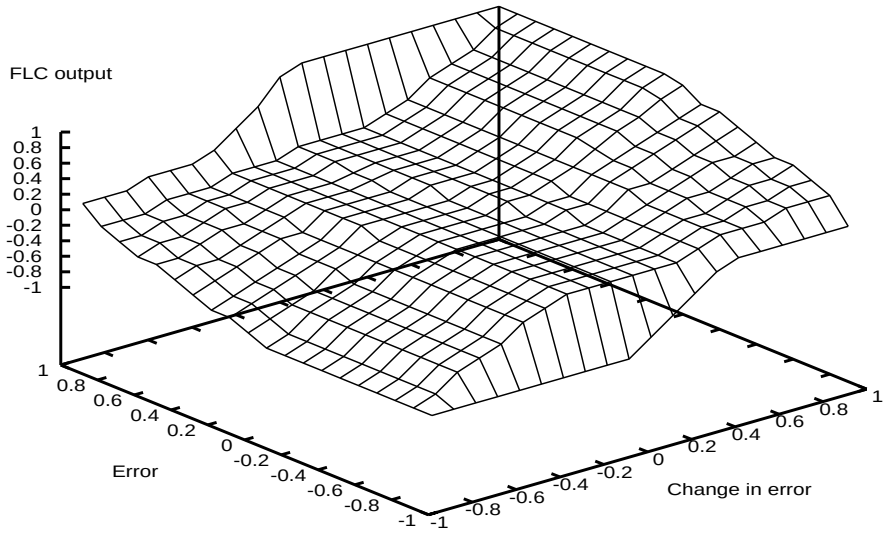


(c) Higher level rule base manually designed

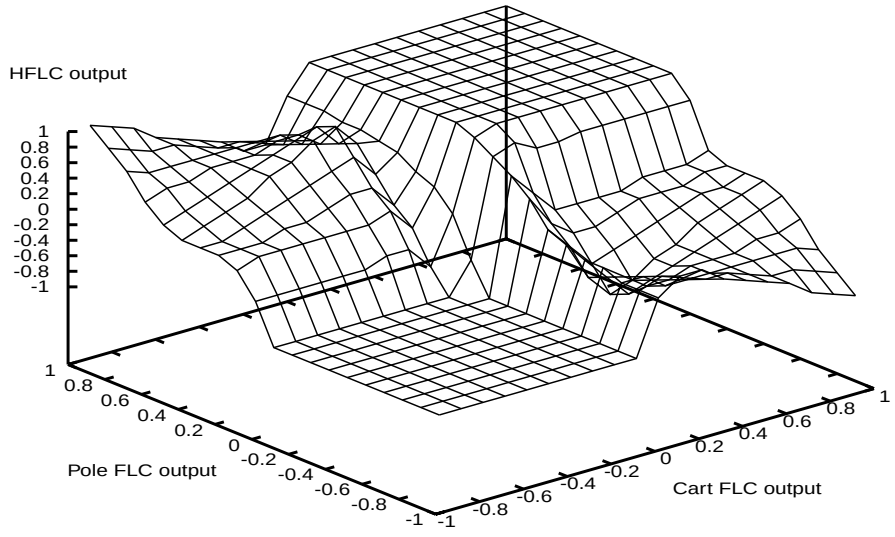
Figure 8: Performance of HFLCs



(a) Pole FLC

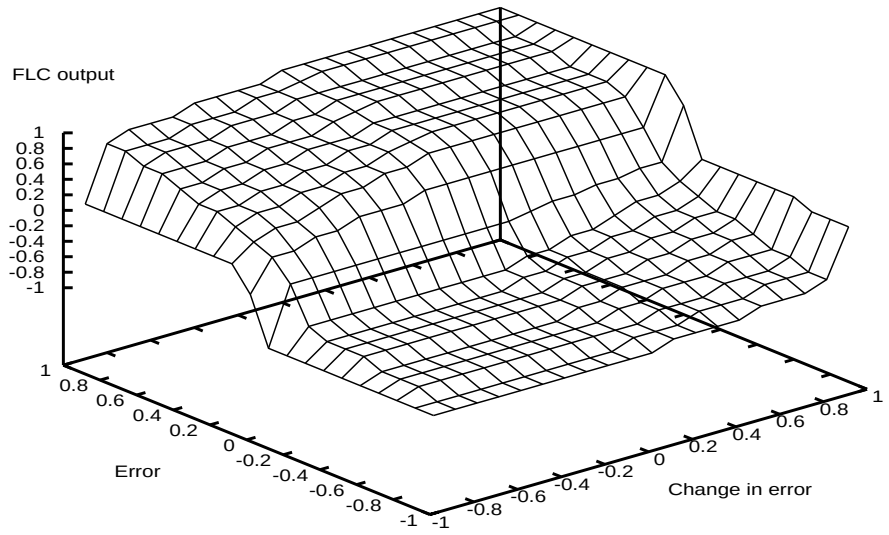


(b) Cart FLC

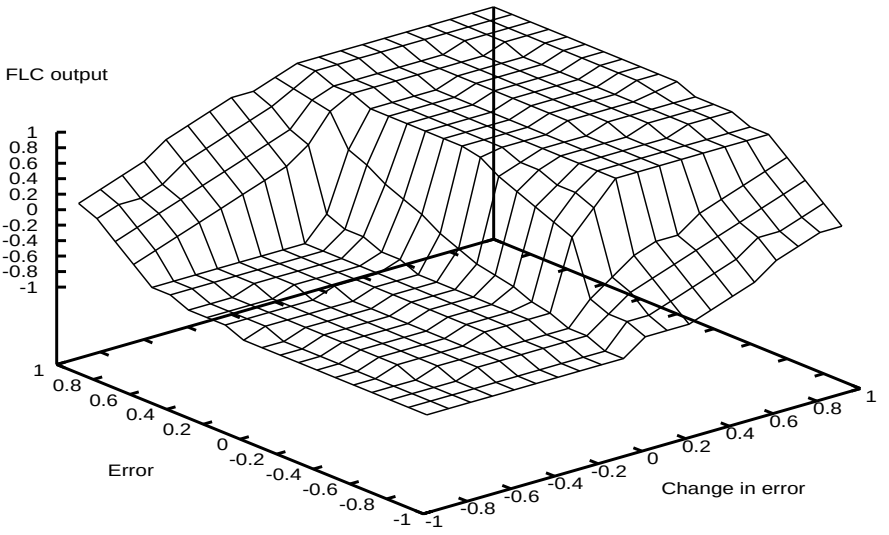


(c) Pole/Cart FLC

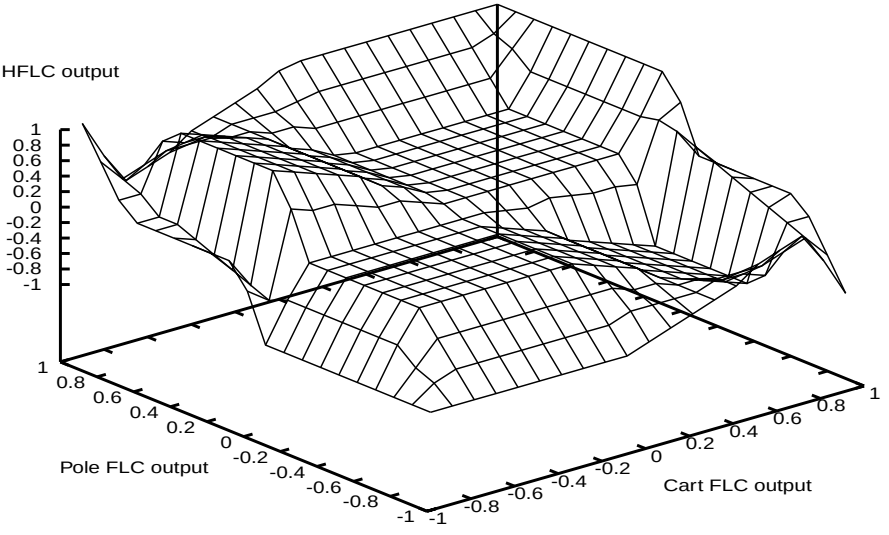
Figure 9: HFLC Control surfaces - Higher level rule base wholly generated



(a) Pole FLC

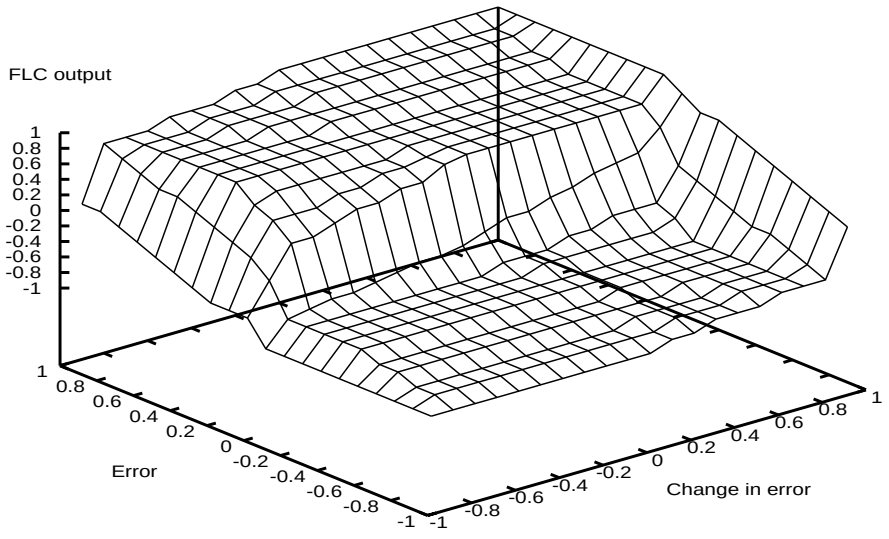


(b) Cart FLC

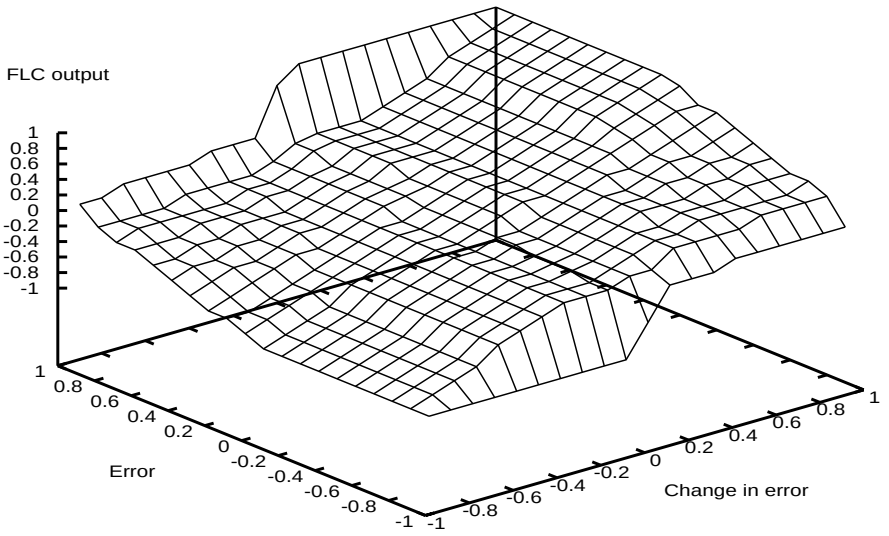


(c) Pole/Cart FLC

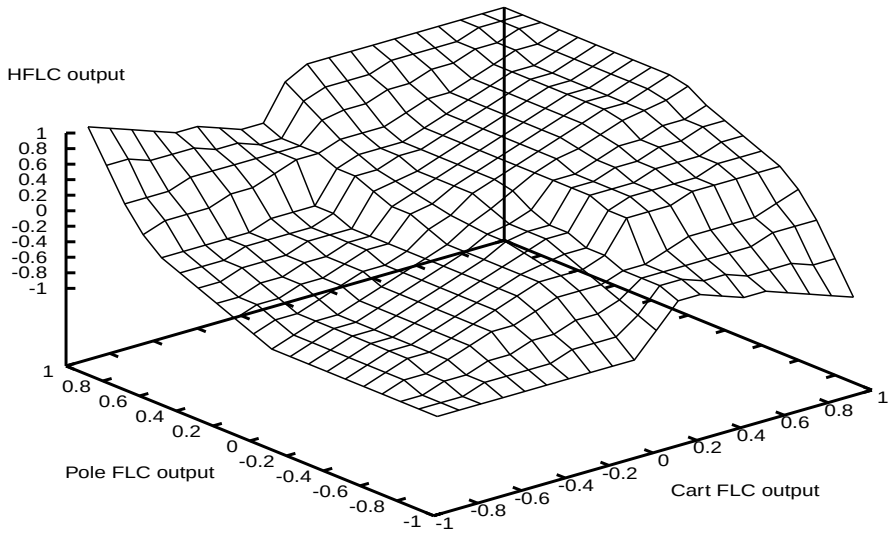
Figure 10: HFLC Control surfaces - Higher level rule base partially generated



(a) Pole FLC



(b) Cart FLC



(c) Pole/Cart FLC

Figure 11: HFLC Control surfaces - Higher level rule base manually designed

6 Evaluating the Work

The cart-pole problem is a classical open-loop control problem. It is a notable example of a highly unstable non-linear problem that is generally used as benchmark problem for the evaluation of new control algorithms and as such it has been well analyzed. In several studies on cart-pole balancing, the system is stabilized within a few seconds or time steps. For example, in [23], the system is stabilized within 5 seconds (or 250 time steps assuming a time step of 0.02 seconds) and in [24] within 7.5 seconds (375 time steps). However, closer inspection of these studies reveal that in [23], the initial cart position was at the center while in [24] the initial positions were a combination of small pole angles (0, 0.26, 0.09, -0.17 degrees) and small cart distances from the center (0,0.3, -0.06, 0.9 metres). Although our HFLC stabilizes the cart-pole within approximately the same time period (300 time steps), the work performed by our HFLC is much harder as some initial conditions are combinations of extreme pole angles ($\pm$15 degrees) and cart positions ($\pm$2.4 metres).

A more meaningful comparison would be against the GENITOR algorithm [25], SANE (Symbiotic, Adaptive Neuro-Evolution) [26] and SEFC (Symbiotic-Evolution-based Fuzzy Controller) [27]. Table 4 shows the number of time steps required to stabilize the cart-pole system from random pole angles ($\pm$15 degrees) and random cart positions ($\pm$2.4 metres) for each method. The number of time steps required to stabilize the system were computed over 50 simulations [27] for each method. From the table, it can be clearly seen that our system outperforms all the methods as our HFLC stabilises the system from any initial position (extreme or not) within 300 time steps.

Method	Time steps to stabilize system			
	Mean	Best	Worst	SD
GENITOR	2,578	415	12,964	2,092
SANE	1,691	46	4,461	984
SEFC	661	10	5,295	749

Table 4: Performance comparisons

In addition to its excellent performance, our method for designing HFLCs has the following merits:

- It can be used to build FLCs with more than two input variables.
- It can be used to build PI- as well as PD-like FLCs. In PI-like FLCs the input variables are the error and its rate of change whereas in PD-like controllers the input variables can be anything else.
- It can be used to build FLCs whose outputs are symmetrical.
- Unlike other methods presented in the literature, the method is complete because the design of the whole HFLC is automated.
- The method is simple because no hybrid techniques are used as in other methods. Differential Evolution does the whole job of automating the design of the controller. Our claim for simplicity is further supported by the use of a highly reduced number of parameters (used for representing the membership functions and the rule base) to be optimized.

7 Conclusions

This paper has presented a method for the automatic design of symmetrical FLCs and HFLCs using Differential Evolution. HFLCs can be used to reduce the number of rules to a linear function of input variables. Designing an HFLC consists of dividing a global task into sub-tasks, designing an independent FLC for each sub-task, and, devising a strategy for coordinating the sub-controllers to achieve the global objective. Symmetricity is a fact that is often overlooked in the literature on FLCs, especially when

automatic design methods are used. Automatic design of symmetrical FLCs cannot be achieved by the naive use of Evolutionary Algorithms; a knowledge of what constitutes a symmetrical FLC is required to harness the power of Evolutionary Algorithms. In particular, symmetrical rule bases and membership functions were defined and a knowledge of symmetricity was exploited to constrain the optimization process.

The method was tested by designing an HFLC using Differential Evolution for controlling an inverted pendulum and it was found out that it performed very well. The experiments carried out showed that the FLCs/HFLCs generated were guaranteed to be symmetrical and they were able to generalize control actions to conditions that were not used during training. Furthermore, the use of a reduced set of parameters to be optimized coupled with the use of a fast Evolutionary Algorithm such as Differential Evolution led to fast optimization i.e. a solution was found within 200 generations using a population of 50 individuals.

The merits of the method are automatic generation of the HFLC and simplicity as the design of the whole HFLC is automated and DE does all the job. No hybrid techniques were used. The highly reduced number of parameters to be optimized (used for representing the membership functions and the rule base) further supports our claim for simplicity.

References

[1] G. Raju, J. Zhou, and R. A. Kisner, "Hierarchical fuzzy control," *International Journal of Control*, vol. 54, no. 5, pp. 1201–1216, 1991.

[2] R. R. Yager, "An alternative procedure for the calculation of fuzzy logic controller values," *J. Japanese Soc. Fuzzy Technol.*, vol. SOFT 3, pp. 736–746, 1991.

[3] R. R. Yager, "On a Hierarchical Structure for Fuzzy Modeling and Control," *IEEE Trans. Syst. Man Cybernet.*, vol. 23, pp. 1189–1197, July/August 1993.

[4] C. Lin, F.-L. Jeng, C.-S. Lee, and R. Raghavan, "Hierarchical Fuzzy logic Water-Level Control in Advanced Boiling Water Reactors," *Nuclear Technology*, vol. 118, pp. 254–262, June 1997.

[5] J. R. Layne and K. M. Passino, "Fuzzy Model Reference Learning Control," in *Proc. 1st IEEE Conf. Contr. Appl.*, (Daytona, Ohio), p. 686, 1992.

[6] H. R. Berenji, Y.-Y. Chen, C.-C. Lee, J.-S. Jang, and S. Murugesan, "A hierarchical approach to designing approximate reasoning-based controllers for dynamic physical systems," in *Uncertainty in Artificial Intelligence 6* (P. P. Bonissone, ed.), pp. 331–343, Amsterdam: Elsevier Science Publishers, 1991.

[7] R. J. Hammell II and T. Sudkamp, "An Adaptive Hierarchical Fuzzy Model," *Expert Systems with Applications*, vol. 11, no. 2, pp. 125–136, 1996.

[8] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst. Man Cybernet.*, vol. 22, pp. 1414–1427, 1992.

[9] P. Horáček and Z. Binder, "Hierarchical Fuzzy Controllers," *Annual Reviews in Control*, vol. 21, pp. 93–101, 1997.

[10] C.-T. Lin and C. S. G. Lee, "Neural-Network-Based Fuzzy Logic Control and Decision System," *IEEE Transactions on Control*, vol. 40, no. 12, pp. 1320–1336, 1991.

[11] W. Pedrycz, *Fuzzy Control and Fuzzy Systems*. New York, NY: Research Studies Press Ltd., John Wiley, 1992.

[12] R. M. Kandadai and J. M. Tien, "A Knowledge-Base Generating Hierarchical Fuzzy-Neural Controller," *IEEE Transactions on Neural Networks*, vol. 8, no. 6, pp. 1531–1541, Nov 1997.

[13] H. R. Berenji and P. Kheddar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Transactions on Neural Networks*, vol. 3, pp. 724–740, 1992.

[14] R. G. G. B. Campello, F. J. Von Zuben, . W. C. Amaral, L. A. C. Meleiro, and R. M. Filho, "Hierarchical fuzzy models within the framework of orthonormal basis functions and their application to bioprocess control," *Chemical Engineering Science*, vol. 58, pp. 4259–4270, 2003.

[15] J. V. Oliveira and J. M. Lemos, "Improving adaptive fuzzy control performance by speeding up identification: Application to an electric furnace," *Journal of Intelligent and Fuzzy Systems*, vol. 6, pp. 297–314, 1998.

[16] P. W. Broomhead and D. Lowe, "Multivariate functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321–355, 1988.

[17] Z.-M. Yeh and K.-H. Li, "A systematic approach for designing multistage fuzzy control systems," *Fuzzy Sets and Systems*, vol. 143, no. 2, pp. 251–273, 2004.

[18] K. Price and R. Storn, "Differential Evolution," *Dr. Dobbs Journal*, pp. 18–24, April 1997.

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

[20] F. Cheong and R. Lai, "On simplifying the automatic design of a fuzzy logic controller," in *Proc. Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS. 2002)*, (New Orleans, Louisiana, USA), 2002.

[21] R. H. Cannon, *Dynamics of Physical Systems*. New York: McGraw Hill, Inc., 1967.

[22] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Problems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-13, pp. 834–846, September/October 1983.

[23] F. H. L. H. K. Lam and P. K. S. Tam, "Design and Stability Analysis of Fuzzy Model-Based Nonlinear Systems Using Genetic Algorithm," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 33, no. 2, pp. 250–257, 2003.

[24] P. T. Chan, A. B. Rad, and K. M. Tsang, "Optimization of Fused Fuzzy Systems via Genetic Algorithms," *IEEE Transactions on Industrial Electronics*, vol. 49, no. 3, pp. 685–692, 2002.

[25] D. Whitley, S. Dominic, R. Das, and C. W. Anderson, "Genetic Reinforcement Learning for Neurocontrol Problems," *Machine Learning*, vol. 13, pp. 259–284, 1993.

[26] D. E. Moriarty and R. Miikkulainen, "Efficient Reinforcement Learning Through Symbiotic Evolution," *Machine Learning*, vol. 22, pp. 11–32, 1996.

[27] C.-F. Juang, J.-Y. Lin, and C.-T. Lin, "Genetic Reinforcement Learning Through Symbiotic Evolution for Fuzzy Controller Design," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 30, no. 2, pp. 290–302, 2000.