

Genetic Algorithm Based Fully Automated and Adaptive Fuzzy Logic Controller

Pintu Chandra Shill[†], Kishore Kumar Pal[‡], Md. Faijul Amin[†], Kazuyuki Murase[†]

[†] Department of System Design Engineering, University of Fukui, 3-9-1 Bunkyo, Fukui-910-8507, Japan

[‡] Khulna University, Khulna, Bangladesh

{pintu,amin,murase}@synapse.his.u-fukui.ac.jp, kkcseku@yahoo.com

Abstract—In this paper, an integration of fuzzy logic controller (FLC) and genetic algorithm (GA) is developed with a view to make the design process fully automatic, without requiring any human expert knowledge. Here, GA is used in two stages simultaneously: the first stage involves selection and definition of fuzzy rules, while the second stage performs an optimal selection of membership function types associated to the fuzzy rules. It is argued that the performance of an FLC greatly depends on the fuzzy rules as well as the types of membership functions associated to the fuzzy sets. Thus, the aforementioned two-stage GA is a viable solution for designing an efficient FLC system. In order to evaluate performance, the proposed approach is applied to a well-known benchmarking controller design task, “backing up a truck reversing system”. The simulation result exhibits superior performance and thereby validates the proposed integrated GA and FLC system.

Index Terms— Fuzzy Logic Controller; Fuzzy Rule Base; Genetic Algorithm; Optimization; Backing up a truck reversing problem

I. INTRODUCTION

Fuzzy logic controllers (FLCs) have numerous practical applications in control [1], [2], prediction [3] and inference [4]. Specifically, they have been used for designing a robust controller that can yield satisfactory performance and deal with uncertainty and imprecision. However, the vast majority of FLCs developed to date are based on the expert knowledge base derived from imprecise heuristic knowledge of experienced control engineers. Such manual approach is very time-consuming as it requires trial and error method to generate fuzzy rules and membership functions (MFs). Moreover, manual design does not guarantee generating optimal or near optimal fuzzy rules and MFs [5], [6]. Clearly, there is a need for developing an automatic FLC design procedure that would not involve any expert knowledge base derived from heuristic knowledge of experienced control operators/engineers.

The main issue in the FLC design includes deciding membership functions' parameters and types, and generating fuzzy rules. Even though all of these are subject to optimization for a better design, many efforts have been directed to the partial optimization, not the entire system. For example, some works optimize the parameters of MFs, while others optimize the fuzzy rules. The parameters of MFs are either decided on the basis of control operator's/person's

experience or are generated automatically. Some methods, such as genetic algorithm (GA), tabu search (TS), particle swarm optimization (PSO) and self-organizing feature map (SOFM) can be used to improve the behavior of this parameter optimization problem [5]-[8], [9], [10].

GA was used by Arslan and Kaya [9] in the determination of membership functions. They applied GA to design a fuzzy logic control system having a single input and output. Bagis [6] proposed a method based on TS algorithm for the determination of membership functions. Bai and Chen [7] proposed an automatic method for students' evaluation task. Its purpose was to automatically construct the grade membership functions of lenient-type grades, strick-type grades, and normal-type grades of fuzzy rules, respectively. The system performs fuzzy reasoning to infer the scores of students based on the constructed grade-membership functions. Meredith et al. [11] presented an approach based on GA to the tuning of membership functions in a FLC for a helicopter. Wong et al. [8] applies PSO to determine appropriate membership functions of the fuzzy system (FS) automatically and presented a motion control structure with a distance fuzzy controller and an angle fuzzy controller for the two-wheeled mobile robot. Yang and Bose [10] presented a method for generating fuzzy membership functions with unsupervised learning using SOFM. The SOFM approach is a two-step procedure: firstly, generate the proper clusters and secondly, generate the fuzzy membership function according to the clusters in the first step. They applied this method to pattern recognition problems.

In this paper, an integrated architecture of FLC and GA has been developed for intelligent control of backing up a truck reversing problem. Here, GA is employed in two simultaneous stages; the first one, referred in this paper as module-1, for the selection and definition of fuzzy rules and the second one, module-2, for the selection of membership function type of each rule. The suggested method can deal with any type of membership functions (i.e., fuzzy sets). The method focuses on finding an optimum set of fuzzy rules and their respective types of membership functions. It is shown that the proposed innovative integrated architecture can generate comprehensible and reliable fuzzy rules and at the same time select the effective type of fuzzy sets (MFs) for each rule by a self-learning adaptive method.

The organization of this paper is as follows: in Section II, we review the literature relevant to the FLCs. In Section III, we provide key ideas of integrated FLC and GA architecture and discuss how GAs can be used to generate a satisfactory fuzzy rule base and to select appropriate types of MFs simultaneously for FLCs. We also show how our proposed approach can be used to solve highly nonlinear complex problems such as backing up a truck control. In Section IV, we explain the problem of backing up a truck in a greater detail. In Section V we present our simulation results and finally section VI presents some concluding remarks based on the present study conclusions and some future directions.

II. FUZZY LOGIC CONTROLLER (FLC)

An FLC comprises four components, which are fuzzifier, knowledge base consisting of rule base (RB) and database (DB), fuzzy inference engine, and defuzzifier as depicted in Fig.1. The FLC works as follows, the crisp inputs are first fuzzified into, in general, input fuzzy sets (however, we will consider only non-singleton fuzzification) which then activate the inference engine and the knowledge base to produce output fuzzy sets. The defuzzifier can then defuzzify fuzzy outputs from the inference engine in order to produce crisp outputs. A brief description of each of the FLC components is given in the following subsections.

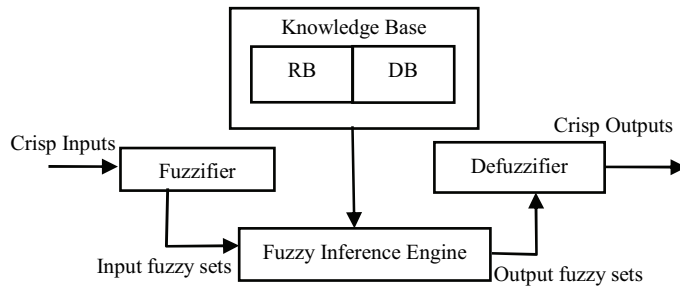


Figure 1. Fuzzy Logic Controller (FLC)

A. Membership functions

Triangular/Trapezoidal MF:

It has the function name *trimf* defined by three points delimiting a triangle.

$$f_{\text{triangulae}}(x) = \begin{cases} 0, & \text{if } x < x_1 \\ 2 \frac{x-x_1}{x_2-x_1}, & \text{if } x_1 \leq x \leq \frac{x_2+x_1}{2} \\ 2 \frac{x_2-x}{x_2-x_1}, & \text{if } \frac{x_2+x_1}{2} < x \leq x_2 \\ 1, & \text{if } x > x_2 \end{cases} \quad (1)$$

Gaussian MF:

Gaussian membership functions are built on the Gaussian distribution curve and are denoted by *gauss_mf*.

$$f_{\text{Gaussian}}(x) = e^{-0.5y^2} \quad \text{where } y = \frac{8(x-x_1)}{x_2-x_1} - 4 \quad (2)$$

B. Fuzzifier

Since the input is in crisp normalized values, a fuzzification operator *fuzz* is used to fuzzify it in fuzzy form. The fuzzifier maps a crisp input vector with p inputs

$x = (x_1, \dots, x_p)^T \in X_1 \times X_2 \times \dots \times X_p \equiv X$ into input fuzzy set, A_x . In Non-singleton fuzzifier: $\mu_A(x_i)$ realizes maximum value 1 at $x = x_i$ and decrease from 1 to 0 while moving away from $x = x_i$ [12]. The fuzzification of the input variable should be realistic. Experience and different procedures as well as different types of MFs should be followed while designing a large fuzzy system for the realistic and accurate output. The wrong fuzzification of the input variable(s) might cause instability and error in the system. Figure 2 shows how fuzzified inputs are calculated from its related different membership forms.

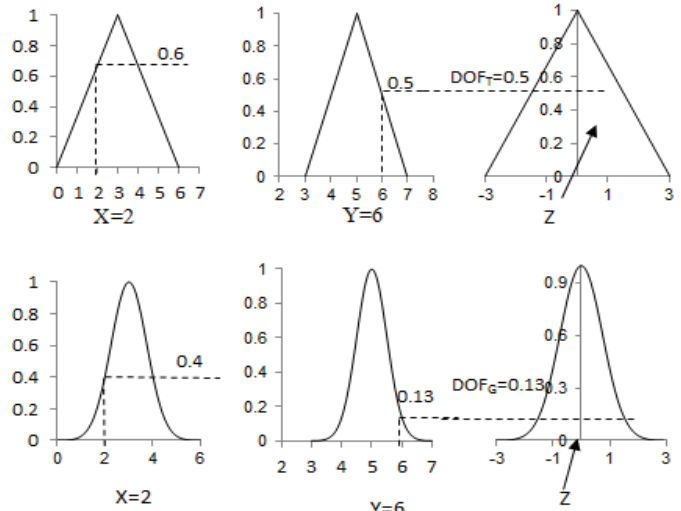


Figure 2. Two different kinds of fuzziness degrees (depends on type of MFs) for a crisp value $X=2$ and $Y=6$

C. Fuzzy Inference Engine

The fuzzy inference engine combines rules and gives a mapping from fuzzy sets in the input universe of discourse $U \in R^n$ to fuzzy sets in the output universe of discourse $V \in R$ based on the fuzzy logic principle. In the inference engine, multiple antecedents in the rules are connected using AND operation, and the degree of membership in the input set sets are combined using those in the output sets using sub-star composition, described in detail in [13]. Multiple rules are then combined by using a join operation.

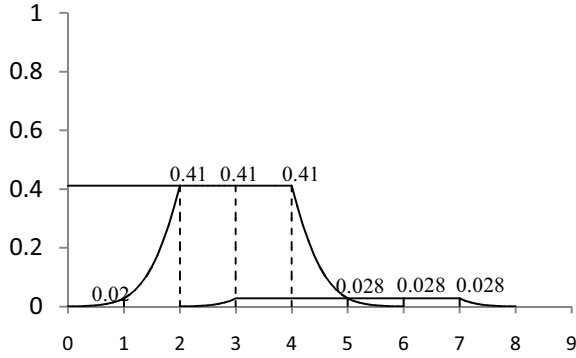
D. Defuzzifier

The defuzzifier performs a mapping from the fuzzy sets in R to crisp points in R . It involves the selection, from the output fuzzy set, the numerical value v^* which can be considered as the best representative element of the fuzzy output set. Maximum defuzzifier produces a numerical value which exhibits the highest degree of membership. The most frequently used center of gravity (COG) defuzzifier takes output as the COG of the fuzzy output. The expression generally used for a discrete fuzzy set is as follows:

$$x_{n+1} = \frac{\int_{X \in U_{n+1}} X \mu_U(X) dX}{\int_{X \in U_{n+1}} \mu_U(X) dX} = \theta^{COG} = \frac{\sum_{j=1}^p \theta_j m_o(\theta_j)}{\sum_{j=1}^p m_o(\theta_j)}$$

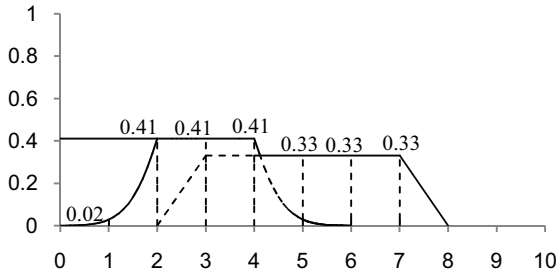
where θ_j denotes the center of gravity (COG) of the consequent label C_j of the rule R_j , $m_o(\theta_j)$ is the firing strength of this rule, and p is the number of rules.

Consider a fuzzy controller with one output variable, and suppose that a given input datum excites two rules of the rule base. Now defuzzification process would yield an output value according to the membership function types associated to linguistic output labels of the two rules. In order to explicate this fact, we consider two cases; gaussian, and mixed MFs. Figure 3 shows that in each case results are different defuzzified value. However, it is easily conceivable that some value would be better than the others. Thus we need a way to define different member function types across the rule base. Some rule may prefer a particular type of MF than the others so as to make an optimal final decision.



(a) Gaussian MFs for FLC

$$x_{n+1} = \frac{1 * 0.028 + 2 * 0.41 + 3 * 0.41 + 4 * 0.41 + 5 * 0.028 + 6 * 0.028 + 7 * 0.028}{0.028 + 0.41 + 0.41 + 0.41 + 0.028 + 0.028 + 0.028} = 3.14$$



(b) Mixed MFs (Gaussian and Triangular) for FLC

$$x_{n+1} = \frac{1 * 0.028 + 2 * 0.41 + 3 * 0.41 + 4 * 0.41 + 5 * 0.33 + 6 * 0.33 + 7 * 0.33}{0.028 + 0.41 + 0.41 + 0.41 + 0.33 + 0.33 + 0.33} = 4.31$$

Figure 3. Effect of defuzzification process for selecting different types of MFs

The value of fuzzified and defuzzified may vary according to the type of MFs (Fig. 2 & 3). In this study, the effect of the

type of the MF on fuzzification and defuzzification process to construct the fuzzy model is investigated along with the selection and definition of fuzzy rules.

E. Fuzzy Rule Base

The general form of a fuzzy rule used in most FLCs is as follows:

IF (a set of condition is satisfied)

THEN (a conclusions can be inferred)

Each fuzzy IF-THEN rule has a premise (or IF) part containing several preconditions and the conclusion (THEN) part describing the output action. The premise and conclusion parts are characterized by some fuzzy sets (or MFs). Suppose that we need to design a multiple-input-multiple-output (MIMO) FLC having p inputs $x_1 \in X_1, \dots, x_p \in X_p$ and c outputs $y_1 \in Y_1, \dots, y_c \in Y_c$ with i -th fuzzy rule of the form:

R_{MIMO}^i : IF x_1 is $F_1^i, \dots$ and x_p is F_p^i , THEN

y_1 is $G_1^i, \dots, y_c$ is G_c^i , $i = 1, \dots, M$

where $F_1^i, \dots, F_p^i$ and $G_1^i, \dots, G_c^i$ are the antecedent and consequent MFs associated with the linguistic p input variables and c output variables, respectively; and M is the number of rules in the rule base.

III. INTEGRATION OF FLCs AND GA

The first step in designing an integrated FLCs and GA architecture, as depicted in Fig. 4, is to decide which parts of the knowledge base are subject to be optimized by the GA. Knowledge base of an FLC system is made up of two components: the parameters of the fuzzy rule base, constituted by the collection of fuzzy rules and MF types associated with the linguistic labels of each fuzzy rule. In this study, both of these two components of knowledge base are adapted using GA.

The integrated FLC and GA architecture consists of following basic steps:

Step1: Identify the variables (inputs, states, outputs) of the plant/problem.

Step 2: Partition the input and output universe of discourses or the interval spanned by each of the variables into a number of fuzzy regions (fuzzy sets) and assign a fuzzy MF i.e., linguistic label to each region.

Step 3: Encode fuzzy regions of the input and output spaces into bit-strings (0 or 1). Encode the MF types using bit-strings as follows: 00, 01, 10 and 11 for the triangular (T), left-triangular (LT), right-triangular (RT), and gaussian (G) respectively.

Step 4: Here, use GA in two stages, module-1 and module-2 (see Fig. 5), as self-learning adaptive methods to generate a set of fuzzy rules and select the MF types, respectively, for each rule.

Step 5: Decode the chromosome in both modules into fuzzy rules and their associative MFs type to determine the performance of a FLC and assign to a fitness value.

In order to clarify the basic ideas of our proposed approach, we have chosen backing up a truck reversing problem having

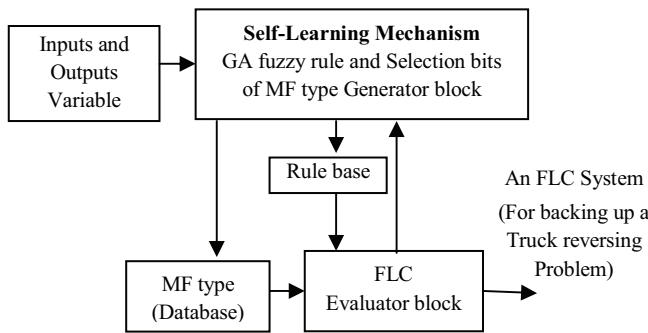


Figure 4. Integrated FLC and GAs Architecture

two inputs x , ϕ and one output θ . It should be mentioned that there can be any number of input and output variables. In order to design an FLC, first divide the range of each variable, the inputs x and ϕ , and the output θ into fuzzy regions and assign a fuzzy MF to each region. We divide the domain regions for x into 5, ϕ into 7, and θ into 7 regions (The usual way to select an odd number between 3 and 9, which is usually same for all the variables). Consequently, there are 35 fuzzy rules (fuzzy rule base or FAM can be formed as a $5 \times (1 \times 7)$ tables with rows to hold the corresponding actions (outputs).

After the division of input output spaces into fuzzy regions, each fuzzy rule can be represented as a bit string and the entire rule base as a table of strings as shown in Table I. Corresponding MFs selection bits are also included in each rule. Before the evolution process of GA the bit string is translated into fuzzy production rules like as Table IV.

To make calculations convenient, the consequent part of each rule and the associated type of MF is automatically set by two parallel GA modules, module-1 and module-2. The FLC generates a fitness value for each rule base according to the following fitness function:

$$\text{Trajectory error/fitness} = \frac{\text{Length of Truck Trajectory}}{\text{Distance(Initial Position, Desired final Position)}}$$

We stop the simulation process and accept the fuzzy rules with respective type of fuzzy sets with $\text{fitness} > \Omega$, where Ω defined as a specified number.

For the integrated architecture, the GA works in the following ways as shown in Fig.5 for graphical illustration:

Step1: Initialization: Some pairs of strings, each pair representing a potential solution, are generated by randomly chosen binary numbers (0 or 1). In each pair, the first string corresponds to a set of fuzzy rules of FLC (GA module-1) and the second string (GA module-2) denotes the types of MFs for corresponding fuzzy rules. Initialize $P=1$, $N=65$, $g=1$ and $G=100$ respectively (Fig. 5).

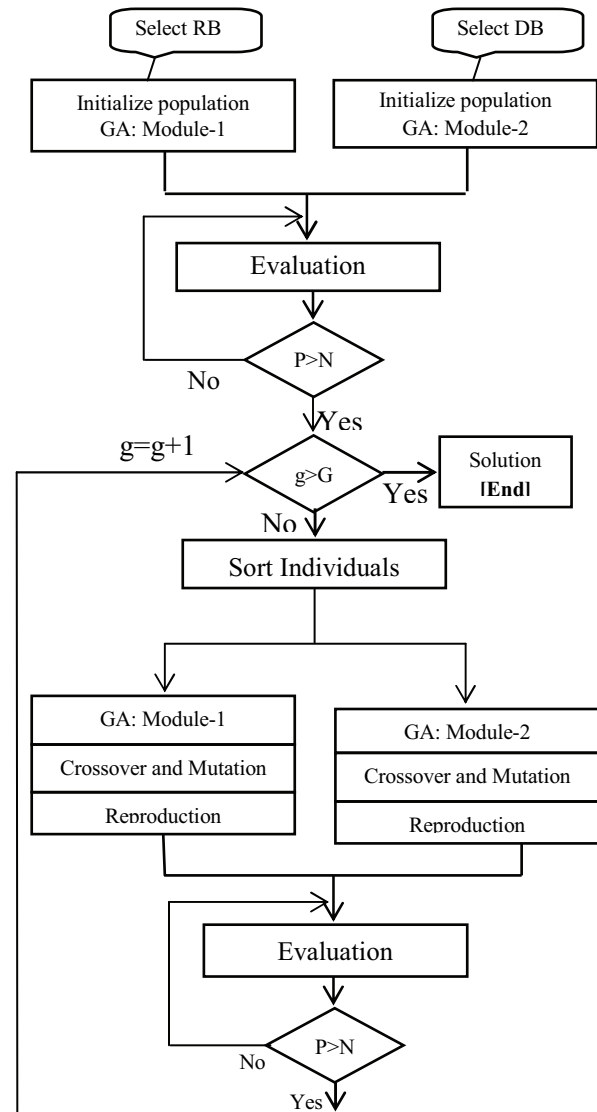


Figure 5. Flow chart for the design of FLC using GAs

Step2: Evaluation: Each potential solution is evaluated and assigned to a fitness value according to the solving performance of the problem. Sort the individuals according to their fitness value.

Step3: Recombination: Apply crossover and mutation operators to individuals of both modules and generate new individuals as well as new generation.

Step4: Evaluation: Evaluate the potential solution and check the termination condition.

Step5: Termination: If termination condition is satisfied then stop and return the best fitted chromosome (set of fuzzy rules of a FLC and membership function types) otherwise go to step3.

TABLE I. OPTIMAL CHROMOSOME OF GA

Input (x)					Input (ϕ)							Output (θ)								MF-Selection bits
LE	LC	CE	RC	RI	RB	RU	RV	VE	LV	LU	LB	NB	NM	NS	ZE	PS	PM	PB		
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	00 (Rule-1)	
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	00 (Rule-2)	
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	01 (Rule-3)	
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	11 (Rule-35)	

It is worth mentioning that the number of parameters such as the size of the population and the probability of applying the genetic operators (i.e., crossover and mutations) influence GAs.

IV. PROBLEM SPECIFICATION: BACKING UP A TRUCK CONTROL

Normal driving instincts can cheat us when attempting to back up a truck to a loading dock [14]. The task is such a difficult one that even for a highly skilled driver needs to go forward and backward numerous times in order to position the truck at the dock successfully. If the driver is not allowed to make forward movements, successful backing becomes improbable. So, the main challenge for the backing up a truck reversing control problem is to design a controller to successfully back up a truck to a loading dock from any initial position.

The problem, to back up a simulated truck to a loading dock in a planner parking lot is depicted in Fig. 6 with simulated geometry. The fuzzy logic control system must find incrementally a path to the loading dock independent of the initial position (x_0, y_0) of the truck. A state vector, consisting of the coordinates of the position of rear center of the truck (x, y) and the angle ϕ of the truck with the horizontal line define the actual position of the truck. The motion of the truck can be described by the following set of equations:

$$\begin{cases} x(t+1) = x(t) + b * \cos(\phi(t)) \\ y(t+1) = y(t) + b * \sin(\phi(t)) \\ \phi(t+1) = \phi(t) + \theta(t) \end{cases} \quad (3)$$

where θ is the steering angle. An FLC receives this state variable as input and calculates the appropriate steering angle $\theta = f(x, \phi)$ that drives the truck toward the goal. The controller have to make the truck reach the loading dock at a right angle ($\phi = 90^\circ$) and to align the position (x, y) of the truck with the desired location (x_f, y_f) . The truck should move backward for some fixed distance b at every stage. Assuming enough clearance between the truck and the loading dock we can ignore the y coordinate at the time of output (steering angle θ) calculation. There is no predefined path for each truck location and therefore the optimum steering angle is unknown. So, at every stage our proposed approach should calculate the steering angle $\theta = f(x, \phi)$ to back up the truck to a loading dock from any position and initial truck angle in the loading zone.

The loading zone corresponds to the plane $\{0, 0\} \times \{100, 100\}$.

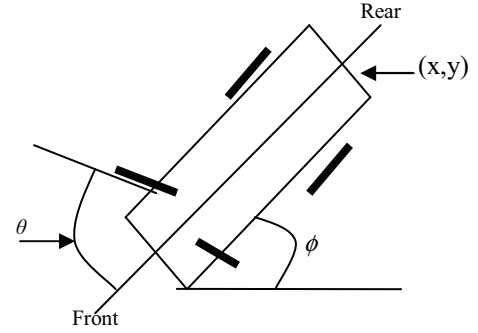


Figure 6. Diagram of Simulated Truck

This experiment is regarded as an example of highly nonlinear complex problems. The fuzzy logic controller must find the correct function that maps points from the three dimensional input-space to the appropriate output variable, continuously from the initial point until the loading dock, and for all combinations of the initial position.

The input and output variable domains are set to be as follows:

$$\begin{aligned} 0 &\leq x \leq 100 \\ -90 &\leq \phi \leq 270 \\ -30 &\leq \theta \leq 30 \end{aligned}$$

V. EXPERIMENTAL RESULTS AND COMPARATIVE ANALYSIS

To evaluate the accuracy of the proposed system, we have carried out a series of experiments in which the controller were evolved in our simulated arena. In this paper fuzzy control rules and MFs types associated to fuzzy control rules are adjusted based on GAs which enables the FLCs to backing up the truck with minimum trajectory error (fitness value).

A. Results

We have obtained the fuzzy control rules from the best chromosomes of GAs (module-1&2) after 100 generations. At the beginning of the simulation, the consequent part of each rule i.e., FAM (Fuzzy Associative Memories) cell is empty. To get the consequent part of each rule in terms of linguistic language form from the developed fuzzy rule generator block, let us start the simulation from thirty initial positions (x_0, y_0, ϕ_0) : $(25, 30, 60^\circ)$, $(80, 30, 150^\circ)$, $(30, 40, 90^\circ)$.. etc.

For all the given initial positions the GA converges to some sets of fuzzy control rules. During the simulation, we generate thirty sets of fuzzy rules (FAM) from thirty different initial positions. Then these rule bases are combined together to form the final FAM shown in table-II, using fuzzy amalgamation (FA) methodology [15, 16]. Table III shows the respective MFs type of fuzzy rules (FAM).

TABLE II. FAM-BANK MATRIX FOR THE FUZZY BACKING UP A TRUCK CONTROLLER

θ		ϕ						
		RB	RU	RV	VE	LV	LU	LB
x	LE	¹ ZE	² PB	³ PM	⁴ NM	⁵ NB	⁶ NB	⁷ NM
	LC	⁸ PM	⁹ PS	¹⁰ NS	¹¹ NM	¹² NB	¹³ PB	¹⁴ NB
	CE	¹⁵ PS	¹⁶ PM	¹⁷ PM	¹⁸ ZE	¹⁹ NS	²⁰ NM	²¹ NS
	RC	²² PB	²³ PB	²⁴ NS	²⁵ PM	²⁶ ZE	²⁷ NS	²⁸ NM
	RI	²⁹ PB	³⁰ PB	³¹ PM	³² PB	³³ PS	³⁴ ZE	³⁵ ZE

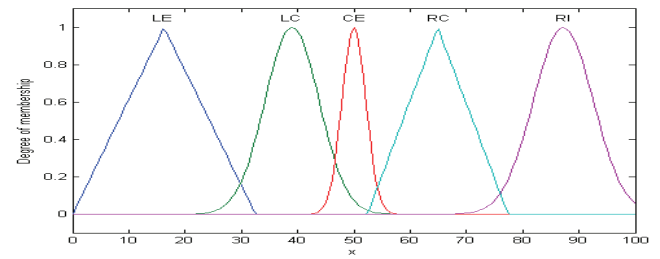
TABLE III. MEMBERSHIP FUNCTION TYPE OF FUZZY RULES(FAM)

θ		ϕ						
		RB	RU	RV	VE	LV	LU	LB
x	LE	¹ T	² T	³ LT	⁴ RT	⁵ G	⁶ T	⁷ G
	LC	⁸ G	⁹ RT	¹⁰ G	¹¹ T	¹² LT	¹³ T	¹⁴ G
	CE	¹⁵ T	¹⁶ LT	¹⁷ G	¹⁸ LT	¹⁹ T	²⁰ T	²¹ G
	RC	²² G	²³ T	²⁴ T	²⁵ G	²⁶ T	²⁷ T	²⁸ G
	RI	²⁹ G	³⁰ T	³¹ T	³² G	³³ T	³⁴ T	³⁵ G

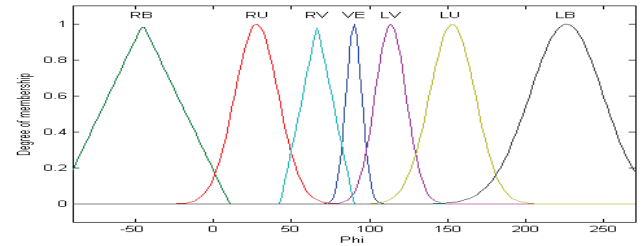
TABLE IV. REPRESENTATION OF FUZZY IF-THEN CONTROL RULES

	GA (Best Chromosome)						
	Antecedents			Consequents			
		Input (X)	Operator	Input (ϕ)		Output (θ)	MFs Type
R-1	If	$x=LE$	and	$\phi=RB$	Then	$\theta=ZE$	Triangular
R-2	If	$x=LE$	and	$\phi=RU$	Then	$\theta=PB$	Triangular
R-3	If	$x=LE$	and	$\phi=RV$	Then	$\theta=PB$	L-Triangular
.	.	.	.	.	.	.	.
R-35	If	$x=RI$	and	$\phi=LB$	Then	$\theta=ZE$	Gaussian

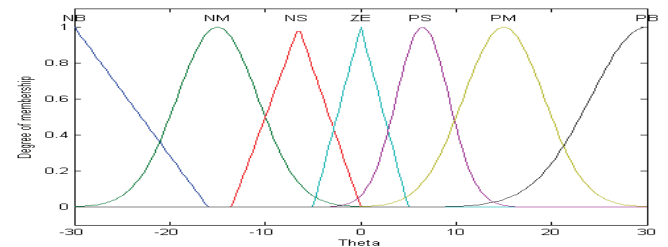
Figure 7 shows (example) the optimal fuzzy sets or membership function of the input and output terms. The MFs and the fuzzy rules specify a particular input-output mapping within a given range. In order to show the complete input-output relationship graphically, when the total number of input and output variables is no more than three, we can construct a control surface. The control surface is a convenient way to examine a two input/one-output control strategy and controller properties. It is obvious that using this useful feature one can more precisely fulfill a quality criterion in full operational range. Figure 8 shows the control surface defined with the 35 fuzzy control rules of Table II.



(a)



(b)



(c)

Figure 7. Variable domain and MFs for (a) x, (b) ϕ and (c) θ

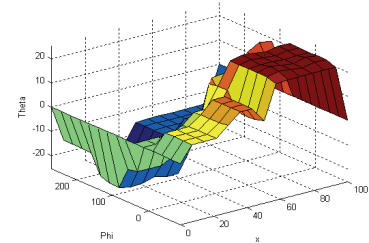
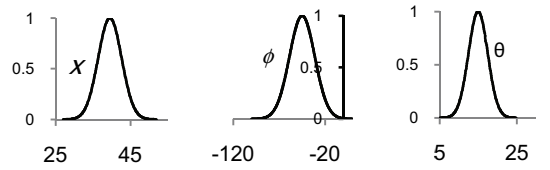
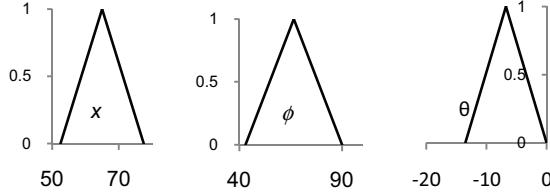


Figure 8. Surface generated by applied FLC

Figure 9 shows the example of two fuzzy control rules with two different types (gaussian and triangular) of MFs. After applying the FAM and their associated types of fuzzy sets (MFs), we obtain the trajectories with trajectory errors by the controller shown in Fig. 10 (a, b) for two different initial positions. The truck should hit the loading dock at a right angle, $\theta_f = 90^\circ$, and the final position at $(x_f, y_f) = (50, 100)$, for any initial state (x_0, y_0, ϕ_0) given to back up. The desired trajectory can be that the truck moves along the target line or coincides with the target line and then goes along y-axis to arrive at the goal.



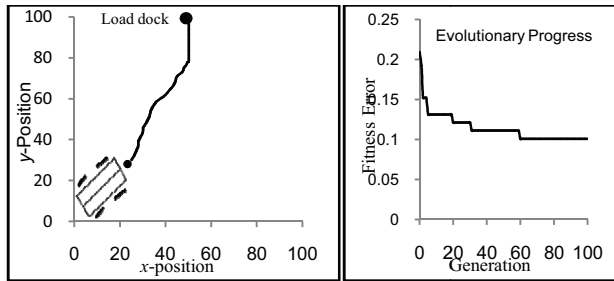
(a) RULE 8: IF $x=LC$ AND $\phi=RB$ THEN $\theta=PM$



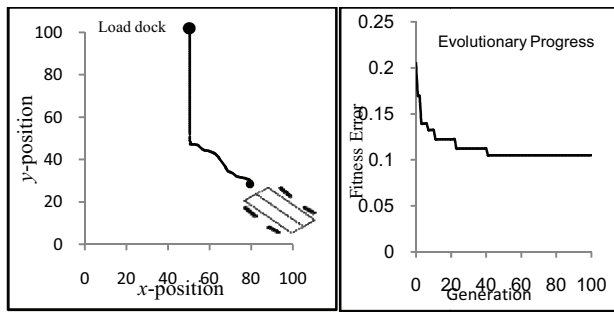
(b) RULE 24: IF $x=RC$ AND $\phi=RV$ THEN $\theta=NS$

Figure 9. Two different MFs form of fuzzy rule (a) Gaussian (b) Triangular

The minimized error after 100 generations for two positions (a, b) are 0.100996 and 0.104841. From Fig. 10 (a) it can be seen that for the initial position (25, 30, 60°) the truck moves towards its goal with comparatively minimum trajectory error because the FLC can make the truck towards the goal position very quickly than that of other initial position, (80, 30, 150°).



(a) Initial position (x_0, y_0, ϕ_0): (25, 30, 60°)



(b) Initial position (x_0, y_0, ϕ_0): (80, 30, 150°)

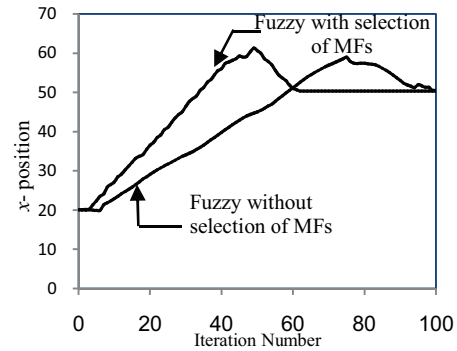
Figure 10. Trajectories and Fitness value of the backing up a truck controlled by the proposed approach.

B. Comparative Analysis: Effect for selecting different type of membership function

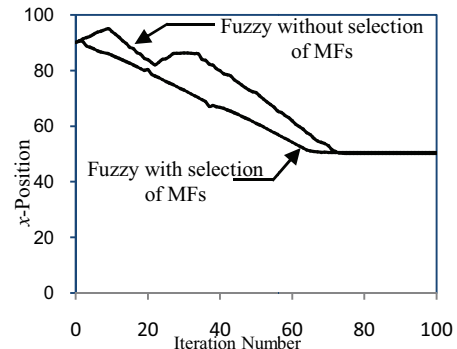
In this section, we investigate the FLC response with selection of different types of MFs and compared FLC response without section of MFs types. Here, we have used

different types of MFs for fuzzy partitions to design an FLC called FLC-1 as shown in Fig. 7. The simple triangular, left-triangular, right-triangular or gaussian membership functions are often used, but they may not be the best choices. Using different MF types decreases the number of trajectory steps and improves the trajectories path that is found by applying only single type of MF. The following (Fig. 11) comparative analysis illustrates this fact.

The FLC-1 is generated by our proposed two-stage GA (module-1&2), while the standard FLC is based on the only single type of MF (triangular/Gaussian). The performance of each controller was observed for backing up a truck reversing problem. It can be seen from Fig. 11 that standard FLC need more time and iteration to converge (meets at $x=50$). On the other hand, FLC-1 reached faster in terms of iteration to converge.



(a)



(b)

Figure 11. Comparison results between fuzzy with SMF and without SMF (a) Case-1: (20, 20, and 80°) (b) Case-2: (90, 10, 110°)

From the result given (Fig. 11), we can see standard FLC and FLC-1 have converged ($x=50$) at iteration 96 and 62, respectively for the case 1 (20, 20, and 80°). FLC-1 also has achieved better precise value at position 50.2635 compare to 51.6439 for standard FLC. The second case (90, 10, 110°) has achieved the same, FLC-1 achieve better convergence and closer to destination than standard FLC and get more precisely than standard FLC. The genetic progress is presented in Fig. 12 which demonstrations the performance of FLC with selection of different type of MFs is better than the response of FLC without selection of type of MFs.

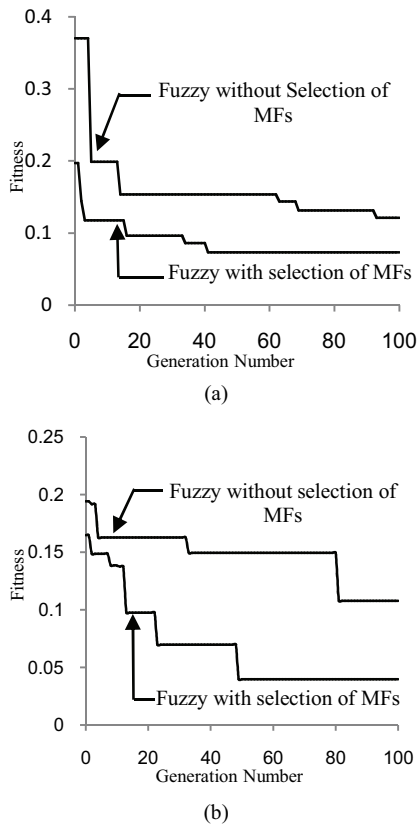


Figure 12. Fitness function value of the solution populations of the methods
(a) Initial position (x_0, y_0, ϕ_0) : $(20, 20, 80^\circ)$ (b) Initial position (x_0, y_0, ϕ_0) :
 $(90, 10, 110^\circ)$

Finally, we have shown that the shape and type of the MFs have a significant influence on the behavior of FLC response. Both the improvement and the deterioration of the systems' performance due to the changes in the MFs have been reported also.

VI. CONCLUSIONS

In this paper, we have found that the innovative integrated FLC and GA architecture is able to construct comprehensible and reliable fuzzy control rules by a self-learning adaptive technique. At the same time, we employed a two-stage GA to select the optimal type of MFs for FLC. The main features of the integrated FLCs and GAs architecture are given below.

The system is an automatic knowledge acquisition method and it is able to learn the fuzzy control rules without any prior knowledge. It also selects optimal types of MFs for FLC. In this way, GAs can build an effective controller relatively quickly. The comprehensive experimental studies also quantify a superb performance of the FLC with selection of different type of MFs when compared with the fuzzy controller without selection of type of MFs. This technique may lead to an increase in the use of FLCs as the previously time-consuming design procedure can be reduced dramatically. The architecture has a self-adaptive learning capability for constructing FLCs.

Suggestions for follow-up work that may come after this paper are as follows. This research work is to be extended for the intelligent control of a mobile robot, truck-trailer controller and control of a robotic arm in the presence of moving or fixed obstacle. This research work is also to be extended to the path planning problem for multiple mobile robots with more than one obstacles either moving or fixed in the workspace.

REFERENCES

- [1] Y. F. Li and C. C. Lan, "Development of fuzzy algorithms for servo systems," *IEEE Contr. Syst. Mag.*, vol. 9, no. 3, pp. 65-73, 1989.
- [2] W. Pedrycz, *Fuzzy Control and Fuzzy Systems*. New York: J. Wiley, 1989.
- [3] C. H. Lee and C. C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 4, pp. 349-366, Aug. 2000.
- [4] T. Yamakawa and T. Koga, "Bio-inspired self-organizing relationship network as knowledge acquisition tool and fuzzy inference engine," in *Proc. IEEE world conf. Comput. Intell.*, 2008, pp. 159-180.
- [5] A. Bagis, "Fuzzy and PD controller based intelligent control of spillway gates of dams," *J. Intell. Fuzzy Syst.*, vol. 14, no. 1, pp. 25-36, 2003.
- [6] A. Bagis, "Determining fuzzy membership functions with tabu search-an application to control," *Fuzzy Sets and Syst.*, vol. 139, pp. 209-25, Oct. 2003.
- [7] S. Bai and S. Chen, "Automatically constructing grade membership functions of fuzzy rules for students' evaluation," *Expert Syst. Appl.*, vol. 35, pp. 1408-1414, Oct. 2008.
- [8] C. C. Wong, H. Y. Wang and S.A. Li, "PSO-based motion fuzzy controller design for mobile robots," *Int. J. Fuzzy Syst.*, vol. 10, no. 1, pp. 24-32, Mar. 2008.
- [9] A. Arslan and M. Kaya, "Determination of fuzzy logic membership functions using genetic algorithms," *Fuzzy Sets and Syst.*, vol. 118, pp. 297-306, Mar. 2001.
- [10] C. C. Yang and N. K. Bose, "Generating fuzzy membership function with self-organizing feature map," *Pattern Recog. Letters*, vol. 27, no. 4, pp. 356-365, Apr. 2006.
- [11] D. L. Meredith, C. L. Karr, and K. K. Kumar, "The use of genetic algorithms in the design of fuzzy logic controllers," in *Proc. third workshop on neural netw.*, vol. 1721, 1992, pp. 549-555.
- [12] J. M. Mendel, *UNCERTAIN Rule-Based Fuzzy Logic Systems*. Engle-wood cliffs, NJ: Prentice-Hall PTR, Jan. 2001.
- [13] L. X. Wang, *Adaptive Fuzzy Systems and Control – Design And Stability Analysis*. NJ: Prentice-Hall PTR, Feb. 1994.
- [14] Li. Ying and Li. Yuanchun, "Neural-fuzzy control of truck backer-upper system using a clustering method," *Neurocomputing*, vol. 70, pp. 680-688, Jan. 2007.
- [15] M. Mohammadian and R. J. Stonier, "Generating fuzzy rules by genetic algorithms," in *Proc. IEEE Int. Works. Robot and Human Comm.*, 1994, pp. 362-367.
- [16] R. J. Stonier, "Evolutionary learning of fuzzy logic controllers over a region of initial states," in *Proc. Congr. Evol. Comput.*, Jul. 1999.
- [17] C. K. Chiang, H. Y. Chung, and Jin-Jye Lin, "A self-learning fuzzy logic controller using genetic algorithms with reinforcements," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 3, Aug. 1997.
- [18] A. K. Nandi and D. K. Pratihari, "Automatic design of fuzzy logic controller using a genetic algorithm to predict power requirement and surface finish in grinding," *J. Materials Processing Tech.*, pp. 288-300, 2004.
- [19] D. H. Kim, K. B. Kim, and E.Y. Cha, "Fuzzy truck control scheme for obstacle avoidance," *J. of Neural Comput. and Appl.*, vol. 18, no. 7, pp. 801-811, Oct. 2009.
- [20] F. Herrera and M. Lozano, "Adaptive genetic operators based on coevolution with fuzzy behaviors," *IEEE Trans. Evol. Comput.*, vol. 5, no. 2, pp. 149-165, Apr. 2001.