Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms

Abdollah Homaifar and Ed McCormick, *Member, IEEE*

***Abstract*— This paper examines the applicability of genetic algorithms (GA's) in the simultaneous design of membership functions and rule sets for fuzzy logic controllers. Previous work using genetic algorithms has focused on the development of rule sets or high performance membership functions; however, the interdependence between these two components suggests a simultaneous design procedure would be a more appropriate methodology. When GA's have been used to develop both, it has been done serially, e.g., design the membership functions and then use them in the design of the rule set. This, however, means that the membership functions were optimized for the initial rule set and not the rule set designed subsequently. GA's are fully capable of creating complete fuzzy controllers given the equations of motion of the system, eliminating the need for human input in the design loop. This new method has been applied to two problems, a cart controller and a truck controller. Beyond the development of these controllers, we also examine the design of a robust controller for the cart problem and its ability to overcome faulty rules.**

I. Introduction

GENETIC algorithms (GA's) are search procedures based on the mechanics of natural selection. They use operations found in natural genetics to guide itself through the paths in the search space. GA's provide a means to search poorly understood, irregular spaces. Because of their robustness, GA's have been successfully applied to a variety of function optimizations, self-adaptive control systems, and learning systems.

Fuzzy systems arose from the desire to describe complex systems with linguistic descriptions [1]. While Boolean systems allow an item to have a membership of either one or zero in a set, fuzzy systems allow for degrees of membership over the range [0, 1]. This imitates the linguistic, nonprecise approach to describing conditions (i.e., cold, very warm) used in everyday life.

Fuzzy controllers allow for a simpler, more human approach to control design and do not demand the mathematical modelling knowledge of more conventional control design methods. As systems become more complex, the ability to describe them mathematically becomes more difficult. For this reason, fuzzy controllers provide reasonable, effective alternatives to classical or state-space controllers.

By using a linguistic approach, fuzzy theory can be integrated into control theory using rules of the form *IF*{condition} *THEN*{action}. Using enough of these rules, one can create a functional controller. In this same way the input variables can be partitioned into overlapping sets which have a linguistic correlation (i.e., cold, warm, hot) to form a membership function. These fuzzy sets are most often triangular in shape but trapezoids and Gaussian functions have also been used. The membership values control the degree to which each rule "fires", illustrating the interdependent relationship between the rule set and the membership functions.

This study investigates the use of genetic algorithms in the design and implementation of fuzzy logic controllers. Previously, generation of membership functions had been a task mainly done either iteratively, by trial-and-error, or by human experts. A task such as this is a natural candidate for a GA since GA's attempt to create membership functions that will cause the controller to perform optimally. In much the same manner, a GA can be used to generate the rules which use these membership functions. Recently, work has been done using GA's to do each of these tasks separately. Karr, for example, has used a GA to generate membership functions for a pH control process [2] and the cart-pole problem [3]. Such work has shown GA's ability to successfully create the individual parts of a fuzzy controller, but since membership functions and rule sets are co-dependent, using a hand-designed rule set with a GA designed membership functions or hand-designed membership functions with a GA designed rule set does not use the GA to its full advantage. Thus, the use of GA's to determine both membership functions and rule sets simultaneously for an optimal or near-optimal controller is the main objective of this work.

Two problems are examined to check the effectiveness of this method. The first is centering and stopping a cart located on a one-dimensional track as described by Thrift [4]. Given an initial velocity and location on the track, the objective is to determine a controller which will bring the cart to zero velocity and zero location in minimum time. Different controllers were designed for this problem by dividing the input and output spaces into different partition sizes.

Manuscript received May 3, 1993; revised May 27, 1994. This work was supported in part by grants from Honeywell Inc. under Grant 48057 and the NASA Center of Research Excellence at North Carolina A&T State University under Grant NAGW-2924.

A. Homaifar is with the Electrical Engineering Department, North Carolina A&T State University, Greensboro, NC 27411 USA.

E. McCormick is with the Center for Digital Systems Engineering, Research Triangle Institute, Research Triangle Park, NC 27709 USA.

IEEE Log Number 9406657.

The second controller comes from the truck-backing system described by Nguyen and Widrow [5] and repeated by Kosko [6]. This controller is used to guide a truck from a given x and y location on a 100×100 grid and at a specific angle to the horizontal (ϕ) to the location of the "loading dock." A controller was designed which performed this task in minimum time, and a comparison was made with the one designed by Kosko.

As cited earlier, GA's have been used in varying degrees to assist in designing the individual parts of a fuzzy controller. In the design of membership functions, GA's have been used not only to determine the base lengths of the fuzzy sets but the location of their peaks also. In this work, a GA has been used to determine only the base lengths of triangular fuzzy sets and not the location of the peaks. Combining this task with the determination of the rule sets has provided proof of a GA's ability to solve this problem.

II. Genetic Algorithms

Genetic algorithms are general purpose optimization algorithms with a probabilistic component that provide a means to search poorly understood, irregular spaces. John Holland originally developed GA's and provided its theoretical foundation in his book, *Adaptation in Natural and Artificial Systems* [7]. Holland developed GA's to simulate some of the processes observed in natural evolution. Evolution is a process that operates on chromosomes (organic devices for encoding the structure of living beings) rather than on living beings. Natural selection links chromosomes with the performance of their decoded structure. The processes of natural selection cause those chromosomes that encode successful structures to reproduce more often than those that do not. Recombination processes create different chromosomes in children by combining material from the chromosomes of the two parents. Mutation may cause the chromosomes of children to be different from those of their parents.

GA's appropriately incorporate these features of natural evolution in computer algorithms to solve difficult problems in the way that nature has done—through evolution. GA's require the problem of maximization (or minimization) to be stated in the form of a cost (objective) function. In a GA, a set of variables for a given problem is encoded into a string (or other coding structure), analogous to a chromosome in nature. Each string, therefore, contains a possible solution to the problem. To determine how well a chromosome solves the problem, it is first broken down into the individual substrings which represent each variable and these values are then used to evaluate the cost function, yielding a "fitness". GA's select parents from a pool of strings (population) according to the basic criteria of "survival of the fittest". It creates new strings by recombining parts of the selected parents in a random manner. In this manner, GA's are able to use historical information as a guide through the search space.

The repopulation of the next generation is done using three methods: reproduction, crossover, and mutation [8]. Through reproduction, strings with high fitnesses receive multiple copies in the next generation while strings with low fitnesses receive fewer copies or even none at all. Crossover refers to taking a string, splitting it into two parts at a randomly generated crossover point and recombining it with another string which has also been split at the same crossover point. This procedure serves to promote change in the best strings which could give them even higher fitnesses. Mutation is the random alteration of a bit in the string which assists in keeping diversity in the population.

GA's work through function evaluation, not through differentiation or other such means. Because of this trait, a GA does not care what type of problem it is asked to maximize, only that it be properly coded. Thus GA's are able to solve a wide range of problems: linear, nonlinear, discontinuous, discrete, etc.

III. Fuzzy Controllers

The development of fuzzy theory came from the inability to describe some physical phenomena with the exact mathematical models dictated by more conventional Boolean models. Fuzziness describes event ambiguity. It measures the degree to which an event occurs, not whether it occurs. The fact that fuzziness is lacking in precision has led to its dismissal by some researchers. Others, however, see fuzzy theory as a powerful tool in the exploration of complex problems because of its ability to determine outputs for a given set of inputs without using a conventional, mathematical model. As Jain notes [9], the basic motivation behind fuzzy set theory is the fact that the conventional methods had become so complex that researchers trying to apply them had to make a choice between a complex system and a complex tool.

Fuzzy theory owes a great deal to human language. As explained by Leung [10], daily languages cannot be precisely characterized on either the syntactic or semantic level. When we speak of temperature in terms such as "hot" or "cold" instead of in physical units such as degrees Fahrenheit or Celsius, we can see language becomes a fuzzy variable whose spatial denotation is imprecise. In this sense, fuzzy theory becomes easily understood because it can be made to resemble a high level language instead of a mathematical language. To describe a universe of discourse, fuzzy sets with names such as "hot" and "cold" are used to create a membership function. By determining the degree of membership of an input in the fuzzy sets of this membership function, one can see the role membership functions play in decoding the linguistic terminology to the values a computer can use. Of course in most respects these membership functions are subjective in nature. What determines the ranges for these fuzzy-set values or the shape of these membership functions? In most cases, membership functions are designed by experts with a knowledge of the system being analyzed. However, human experts cannot be expected to provide optimal membership functions for a given system. Often, these functions are modified iteratively while trying to obtain optimality.

How are these membership functions used in fuzzy controllers? In its simplest form a fuzzy logic controller is simply a set of rules describing a set of actions to be taken for a given set of inputs. It is easiest to think of these rules as if-

then statements of the form *IF*{set of inputs} *THEN*{outputs}. As an example, consider a fuzzy controller used in the cart controller problem. One rule might be *IF*{distance from 0 very far in positive x-direction} and {velocity $\gg$ 0) *THEN*{apply a force $\ll 0$}. Another rule may be *IF*{distance from 0 near in negative x-direction} and (velocity = 0) *THEN*{apply a force > 0). Since "very far" applies to a range of distances which also may belong to another fuzzy-set variable (i.e., "far") which has rules of its own, the output which results from "defuzzification" of the application of these rules must take into account how much each rule applies before determining how much output must be applied. Usually a centroid method is used to account for the influence of each rule on the output.

IV. Algorithm Description

The basis for the software used in this paper is the Simple Genetic Algorithm (SGA) program developed by Goldberg [8]. The SGA program allows the user to define the values for population size, maximum number of generations, probability of crossover, and probability of mutation. Their respective values are 100, 100, 0.7, and 0.03. In order to select the individuals for the next generation, tournament selection was used instead of SGA's roulette wheel selection. In tournament selection, two or more members of the population are selected at random and their fitness compared. The member with the highest fitness advances to the next generation.

The Simple Genetic Algorithm uses binary strings to encode the parameters which are to be optimized. While this method could also be used in the determination of the fuzzy controller design, a more representative method was chosen. To illustrate this method, consider the cart centering problem. First, the number of alleles (individual locations which make up the string) was determined from the size of the rule set plus the number of fuzzy sets used to partition the spaces of the input and output variables. In the cart problem, the bases of the triangles which formed the output space were fixed to ease the computational burden (reduction of string length and simplification of the defuzzification process), while the input variables, x-location and velocity, had base lengths that were determined by the GA. In this example five triangular fuzzy sets were used to partition the input and output spaces: negative medium (NM), negative small (NS), zero (ZE), positive small (PS), and positive medium (PM). The rule set, then, contains twenty-five (5×5) rules to account for every possible combination of input fuzzy sets. The rules are of the form, *IF*(x is {NM, NS, ZE, PS, or PM}) and (v is {NM, NS, ZE, PS, or PM}) *THEN*{*output*}, where *output* is one of the fuzzy sets used to partition the output space. The two input spaces use a total of ten triangles, so the string to represent a given rule set and membership function combination would have thirty-five alleles (25 + 10). No additional alleles are needed for the output triangles because their base lengths are fixed. Note that the term alleles is used instead of bits, because the value of each location in the string contains either the number of the output fuzzy set to be used for a given rule (the first twenty-five alleles where NM = 1, NS = 2, etc.) or the value which will be converted to the length of the base

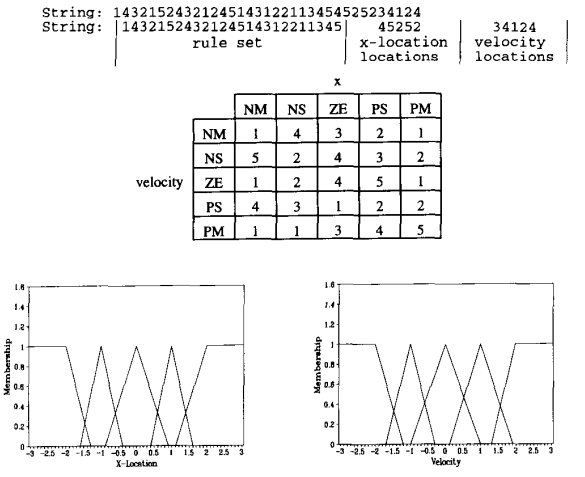

Fig. 1. Example of string fuzzy controller conversion.

of the triangles which make up the input spaces (the last ten alleles). The calculation of the triangle bases from the allele values (1–5) were done as follows (the values are specific to the cart-centering controller described above):

1) Subtract 1 from the allele value and divide by 10 (making the range now 0–0.4).
2) Subtract this value from one (which is the fixed distance between the peaks of each triangle).
3) Doubling this value gives the base length for each particular triangle. This value can be anywhere from 1.2 m to 2.0 m. A base length of 2.0 m means the end point of the triangle extends to the peak of the surrounding triangle while if two adjacent triangles had the smallest possible base lengths (1.2 m), they would be assured of having a 0.2 m overlap.

Following this method, the string representing the controller is integer-based instead of binary-based. The alleles representing the rule sets have values in the set $\{1, 2, \cdots,$ Number of Output Sets$\}$ while the alleles representing the bases of the triangles are in the set $\{1, 2, \cdots, 5\}$. This change in structure does not change the basic functioning of the GA. In fact, the only difference occurs within mutation, since an allele was allowed to change into any value other than its present value. Thus, the two main ingredients of a fuzzy controller, the rule set and the membership functions, are incorporated into a single string which the GA will seek to optimize. An example of how each string is broken down is given in Fig. 1.

A problem which comes to light in the optimization is the use of mathematical models to evaluate the fitness of a given string. One of the strong points of fuzzy controllers is the fact that they do not require mathematical models. However, to obtain a fitness for a given controller, the GA must have a method to evaluate the controller's performance. In this sense we have negated one of the fuzzy controllers advantages in order to use the power of GA's to optimize this controller.

V. Applicability of GAs to Fuzzy Controllers

The application of genetic algorithms to fuzzy logic controllers holds a great deal of promise in overcoming two of

the major problems in fuzzy controller design, design time and design optimality. Previous work has been done mainly in two areas: learning the fuzzy rules and learning membership functions. A GA's robustness enables it to cover a complex search space in a relatively short period of time while ensuring an optimal or near-optimal solution. Because of this capability, GA's are a natural match for fuzzy controllers.

Thrift's paper [4] examines the feasibility of using GA's to find fuzzy rules. In this paper, fuzzy control synthesis is done in decision table form. The problem examined is centering a cart of mass m on a one dimensional track. The objective is to move the cart from a given initial position and velocity to zero position and velocity in minimum time. This is done through the application of a force **F** from the controller. For 100 runs with random starting points, the average number of time steps for the hand-designed fuzzy controller to bring the cart to zero position and velocity was 164. In comparison, a GA designed controller using the same starting points had an average of 143 time steps. As Thrift notes, while the GA based fuzzy rules perform reasonably well, work could be done to further improve its performance, such as letting the GA determine the endpoints of the membership functions.

Karr [3] examined the feasability of using a GA to find high performance membership functions for a controller for a pole-cart system. The task for the controller is as follows:

> A wheeled cart has a rigid pole hinged to its top. The cart is free to move right or left along a straight bounded track and the pole is free to move within the vertical plane parallel to the track. The cart is to be kept within the predefined limits of the track and the pole should be prevented from falling beyond a predefined vertical angle by applying a force of fixed magnitude to the left or right of the base of the cart.

The objective is to bring the cart to rest at the center of the track with the pole balanced, much the same as in Thrift's paper. Also, he examined the use of micro-GA, a small population GA developed by Krishnakumar [11], to determine an adaptive real-time controller for the same problem where system parameters may be time varying. In determining the membership functions, the GA was used to determine the anchor points for each of the linguistic variables used. In the nonadaptive problem, the GA designed fuzzy logic controller consistently outperformed the controller he had hand-designed.

For the adaptive controller, the micro-GA controller had the best performance. The nonadaptive author-designed controller always became unstable while being used in the adaptive case while the nonadaptive GA controller and the micro-GA designed adaptive controller were always able to complete the task. The difference between the two GA designed controllers was in their convergence times; the micro-GA controller consistently balanced the system faster than the nonadaptive GA controller, whose performance was deemed unacceptable.

Others have sought to optimize fuzzy controllers using other methods. For example, Procyk and Mamdani [12] iteratively designed membership functions. While all these methodologies have provided improvements in fuzzy controller design, they have a major limitation; how can an optimal design be obtained when one of the two main components is designed using a nonoptimizing method. Logically, to obtain an optimal rule set and set of membership functions, the two must be designed together so the links between them can be fully exploited. This process is now beginning to be explored by researchers. Previous work done by Homaifar and McCormick [13] examined the initial applicability of GA to solving the cart-centering problem and laid the foundation for this more in-depth study. Also, Nomura *et al.* [14], examined using a GA to determine both the membership function and optimum number of rules for a single input, single output nonlinear system. Since the two systems had only one input, the number of fuzzy sets for that input was also the number of rules. They used a binary string structure for the GA where each bit location mapped to a corresponding location in the input space. If a bit representing a given location in the input space were a one, that meant there was the peak of a fuzzy set located there. While able to successfully optimize both the membership function and the rule set, there may be a disadvantage in the fact that the endpoints of a given fuzzy set were always located at the peaks of the adjacent fuzzy sets. This may not necessarily be the case in an optimal membership function. Lee and Takagi [15] have also used GA to approach simultaneous membership function and rule set design. They developed a four rule controller which was able to control the inverted pendulum problem. The GA was given more flexibility in designing the rule set and membership functions for this difficult nonlinear problem. This flexibility, however, led to solving for 360 parameters which was encoded into strings 2880 bits long! Additional information concerning those doing work in this area can be found in papers by Nishiyama *et al.* [16], Qian *et al.* [17], and Tsuchiya *et al.* [18]. These examples show that by using GA's to design both simultaneously, the two elements of fuzzy controllers can be fully integrated to deliver a more finely tuned, high performance controller.

VI. Cart-Centering Problem

A common problem used in literature is the centering of a cart of mass m, on a one-dimensional track. The input variables for this problem are the cart's location on the track, x, and the cart's velocity, v. The objective is to find a controller which can provide a force F which will bring the cart to $x = 0$ and $v = 0$ from an arbitrary initial condition (x_0 and v_0) in minimum time. The equations of motion for the cart are

$$x(t+\tau) = x(t) + \tau v(t)$$
$$v(t+\tau) = v(t) + \tau \frac{F(t)}{m}$$

where τ is the time step. The values for the constants and the ranges for the variables are given in Table I.

Three controllers were developed for the cart-centering problem. They will be referred to by the number of fuzzy sets that partition the x-location, velocity, and output. For example, the controller which had the x-location divided into five fuzzy sets, the velocity divided into five fuzzy sets, and the output divided into seven fuzzy sets was called the 557 controller. In all cases, the locations of the peaks of the triangles forming

TABLE I
CONSTANTS AND RANGES FOR CART PROBLEM

Variable or	Value
m	20 kg
τ	.02 sec
x	-2 to +2 m
v	-2 to +2 m/s
F	-150 to +150 N

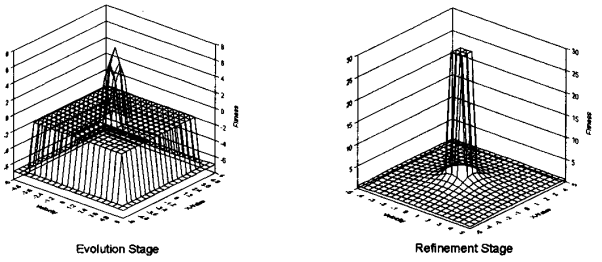

Fig. 2. Evolution and refinement stage fitness functions for cart controller.

the fuzzy sets were evenly spaced. For example, if the x-location was partitioned into five fuzzy sets, the peaks were at $-2, -1, 0, 1$, and 2. As stated earlier, GA was allowed to determine the location of the triangle bases for the input variables only, while the output fuzzy set locations were fixed.

VII. TRUCK BACKING SYSTEM

The truck backing problem consists of a truck located somewhere on a 100 × 100 grid at a given angle to the horizontal, ϕ. The objective is to take the truck in minimum time from this arbitrary initial condition (x_i, y_i, ϕ_i) to the location of the "loading dock" (x = 50 m, y = 100 m) while making the truck vertical to the dock (ϕ =90 degrees). The controller will provide a turning angle, θ, that moves the wheels and in turn the truck every time step. The equations of motion for the truck are given as

$$\phi' = \phi + \theta$$
$$x' = x + r\cos(\phi')$$
$$y' = y + r\sin(\phi')$$

where r is the fixed distance the truck backs each time step, and ϕ', x', and y' are, respectively, the new truck angle, x-location, and y-location. The problem is illustrated in Fig. 2. The values for the constants and the ranges for the variables are given in Table II. Although y-location could be considered a variable, it was assumed in this study that the truck was sufficiently far from the loading dock in the y-direction that the y-distance could be ignored. This assumption assists in simplicity and is consistent with work the work first done by Nguyen and Widrow [5] and later by Wang and Mendel [16].

The truck-backing system created a number of difficulties that were not present in the cart controller. One of the first differences encountered was the realization that instead of the rule set being a flat matrix, it was more like a matrix wrapped around a cylinder. This came from the fact that the truck angle, ϕ, was allowed to vary from -90 degrees to +270 degrees, a circle. Therefore, the two fuzzy sets that

TABLE II
CONSTANTS AND RANGES FOR TRUCK-BACKING PROBLEM

Variable or	Value
r	2m
θ	-30° to +30°
ϕ	-90°to +270°
x	0m to 100m

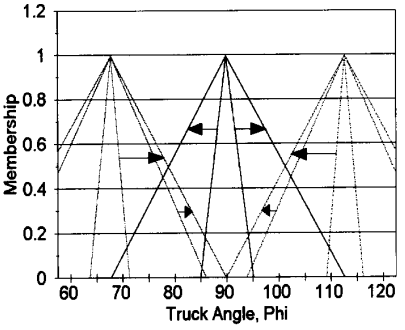

Fig. 3. Possible base lengths in truck controller problem.

contained -90 degrees and +270 degrees must also be attached meaning they wrapped around to form a cylinder as would happen if the top and bottom of the rule set matrix were connected together. This led to the necessity to dictate to the GA more exactly how it should choose the locations for the triangle endpoints making up ϕ's membership function. In part because of this complication and because of the desire to allow the GA more flexibility in determining the endpoints, a new method was developed for this part of the problem. In solving the cart problem for the 555 controller, there were five possible base lengths of the fuzzy partitions: 1.2, 1.4, 1.6, 1.8, and 2.0. For the truck-backing problem, however, the GA was modified so the incrementation for a given triangle depended on the location of the endpoint of the adjacent triangle. This process allowed for a much greater variation of the endpoints and ultimately better results. This is illustrated in Fig. 3 where the ranges of possible base lengths for the outside membership functions varies according to the base length of the center fuzzy partition. Instead of fixed points, the resolution of the possible base lengths changed according to the adjacent triangle. Also included in this problem was the determination of the output's membership function by the GA. This added additional alleles to the GA strings and made complicated output centroid calculation more complex since the centroid of a given output fuzzy set could no longer be assumed to reside at the location of its peak.

VIII. RESULTS FOR CART CONTROLLER

A. Initial Conditions

To find a satisfactory controller, the controller must be able to operate over the entire range of the input spaces. For a GA to properly design fuzzy controllers, this fact must be

TABLE III
INITIAL CONDITIONS FOR CART CONTROLLER

Controller	Initial Conditions (x_0, v_0)
333	(-2,-2) (-2,2) (0,0) (2,-2) (2,2)
555	(-2,-2) (-2,2) (-1,-1) (-1,1) (0,0) (1,-1) (1,1) (2,-2) (2,2)
557	(-2,-2) (-2,2) (-1,-1) (-1,1) (0,0) (1,-1) (1,1) (2,-2) (2,2)

integrated into the function evaluation. This was done by using multiple initial conditions in the evaluation of each member of the population. If a single initial condition were used, for example $x_0 = 0.7$ m and $v_0 = -0.5$ m/s, then the GA would find a controller which would work well around that particular point but may fail elsewhere. This makes the choice of initial conditions an important consideration. The points must be chosen to sufficiently cover the input spaces, but at the same time, more initial conditions leads to increased run time for the program. This is done in much the same manner as Karr's use of four initial conditions in his pole-cart balancing fuzzy controller [6]. These initial conditions are listed in Table III. In evaluating each member of the population, the total fitness of the individual was the sum of the fitnesses at each initial condition.

B. Fitness Function

The fitness function proved to be the most challenging aspect of applying GA's to fuzzy controller design. The process was divided into two stages, an evolution stage and a refinement stage. In the evolution stage, the GA was used to find satisfactory controllers, while in the refinement stage, the GA used the previously developed controllers and attempted to minimize the amount of time needed to bring both x-location and velocity to zero.

For the first stage, which lasted through generation 30, the fitness function rewarded a member of the population according to how well it came to the tolerance value, ± 0.5 for both x-location (x) and velocity (v). These tolerance values provided sufficient proof that the controller was heading in the correct direction and slowing down as it approached zero velocity and x-location. As the absolute value of both x and velocity become smaller, the steady-state rule ensures that they will continue decreasing. The generation at which the evolution stage ended, 30, was obtained experimentally by examining various runs and monitoring when the population average began to level off as it approaches the population maximum. This gives a relatively good assurance that the majority of the population can successfully complete the task. This fitness function is shown in Appendix A. If the controller succeeded in bringing x and v within the tolerance, it was given a fitness relative to the time it took. The minimum fitness the controller could have if it met the tolerances was 8 (since 175 was the time limit). If the controller "timed out" (i.e., did not converge by 175 time steps), it was either slightly punished with a negative fitness or slightly rewarded depending on x-location and velocity (the maximum fitness for this condition could only be 3.5 if one tolerance were 0 and the other 1). If the controller diverged (i.e., x or v were greater than 5.0), the fitness was given a larger negative value. These values helped ensure that the GA was rewarded for controllers that worked or nearly worked and punished for those that didn't. Through the use of this reinforcement/reward scheme, the GA was able to develop controllers which could solve all the initial conditions.

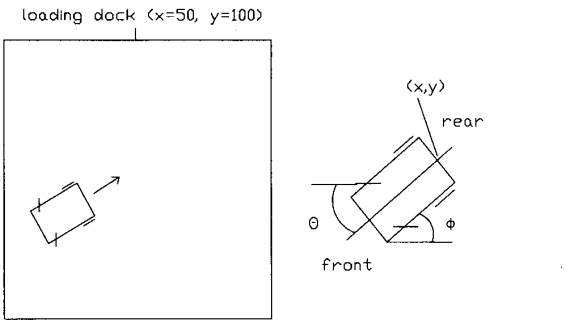

Fig. 4. Truck and loading dock illustration.

The second stage, from generation 31 to generation 100, was based almost completely on time. If the controller reached the tolerance values it was rewarded according to how short a time it took. If the controller "timed out," it was punished according to how much it missed the tolerance values, and if the controller diverged, it was given a very large negative fitness which would probably ensure its failure to continue on to the next generation. This fitness function is also shown in Appendix A.

As was done for the determination for the crossover from evolution to refinement stage, the fitness functions were obtained throughout experimentation. The overriding qualification for the evolution phase was to find controllers that worked, no matter how long they took. Examining the fitness function shows that the minimum fitness for a successful completion of an initial condition is eight, while a time out within relaxed tolerances yields a maximum fitness of 3.5 (say $x = 0$ and velocity -1). If a timeout occurs and the controller has not come within tolerance it is slightly punished as opposed to the stronger punishment it receives if it diverges. Much the same occurs in the refinement stage where the minimum fitness of a successful run is zero (time = 175), while the minimum punishment for not coming within tolerance values is -21 and the maximum punishment is -300. These fitness functions are shown in Fig. 4. The evolution stage figure assumes the minimum possible fitness mentioned earlier. In the refinement stage graph, the maximum fitness, 30, comes from an assumption of 75 time steps for a given initial condition to complete the simulation successfully. These values are for illustrative purposes only as the completion times had a large variation. Also, since the refinement stage graph shows x and velocity values from -5 to +5, the -300 penalty value is not illustrated. As the work progressed with the cart controller, it was decided that the tolerance values were too great and would be reduced as the truck controller was developed. This was reasonable since the cart controller was used to develop the premise of dual rule set and membership function optimization, while the truck controller would be more elaborate proof of the concept.

velocity \ x	NM	NS	ZE	PS	PM
NM	5	4	4	4	2
NS	5	5	5	1	1
ZE	5	5	3	1	1
PS	5	5	1	1	1
PM	4	2	2	2	1

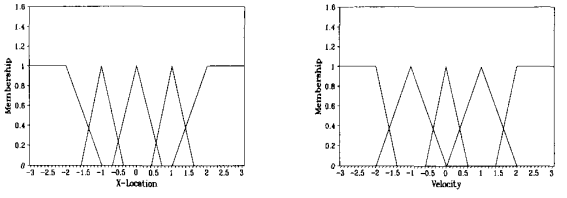

Fig. 5. 555 fuzzy cart controller.

C. 555 Controller

In this section we present one of the controllers developed using GA's. The 555 controller used 25 rules and needed 10 alleles to determine the location of the bases of the fuzzy sets covering the input spaces, giving a total string length of 35. Considering that each allele could have a value between one and five, this meant that if a binary string had been used, three bits would be necessary to represent the same information. This would yield a string length of 105. Using this value for string length would indicate the need for a much larger population than the one being used to ensure enough initial population diversity. Therefore, we cannot be assured that the GA will converge to an optimal or near-optimal solution. Even using the relatively small population size of 100, run times took between two and one-half to three hours. However, the performance of the controller did indicate that the GA was finding a near-optimal controller, when compared with solutions such as the one by Thrift [4]. Fig. 5 shows the resultant best controller determined by GA.

D. Comparison of Cart Controllers

Once the GA-designed controllers had been determined, a study was made to determine their performance and stability. Because there is no accepted method to determine stability for fuzzy controllers as there is for classical and state-space controllers, a "brute force" method was adopted. To examine the controllers, the input space of each variable was divided into 40 points. Then each point was examined, one by one, to determine if the controller diverged anywhere in the input space. For 40 points in the x-location space and 40 points in the velocity space, this yields a total of 1600 points. While the ability of a controller to satisfy all these points does not necessarily guarantee its stability (since it only takes one point to make a controller unstable), this did ensure some measure of confidence in the procedure. While examining each point, it was a simple task to also count the number of time steps used to bring all these points within the tolerance values, and these numbers are given in Table IV. Also shown in Table IV is the 555* controller, which was designed with the membership functions fixed; a GA was used to design only a rule set, with the membership function being done by hand. This controller, shown in Fig. 6, was created for comparison purposes to illustrate the importance of membership function selection. As Table IV shows all the controllers were able to successfully bring the system within the tolerance values.

TABLE IV
COMPARISON OF CART CONTROLLERS

	333	555	557	555*
No. of Initial Conditions	1600	1600	1600	1600
Time Steps Required	82380	68208	69756	90156
Avg. No. of Time Steps	51.49	42.63	43.60	56.35
% Difference w/555	20.8%	—	2.3%	32.2%
No. of Failures	0	0	0	0

velocity \ x	NM	NS	ZE	PS	PM
NM	5	5	5	3	2
NS	5	5	5	1	1
ZE	5	5	3	1	1
PS	5	5	1	1	1
PM	5	1	4	2	1

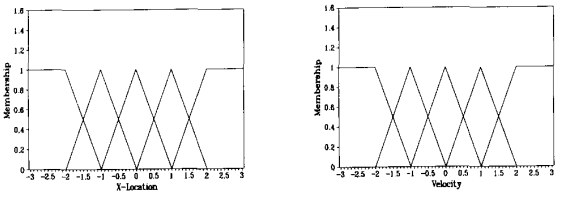

Fig. 6. 555* fuzzy cart controller using fixed membership functions.

Table IV shows that the best performance, on average, came from the 555 controller. The 333 controller, while being the simplest, did not have the flexibility to produce fast response times. Also, the 557 controller, which had smaller partitions of the output space, performed nearly as well as the 555 controller. Finally, note that while the 555* controller was able to bring the cart to equilibrium for all points, its performance was clearly inferior, needing almost 1/3 longer than the GA designed rule set and membership function combination, showing the importance of proper membership function design.

E. Robustness Examination of 555 Controller

So far we have developed a controller which is able to effectively bring a cart of mass 20 kg to equilibrium. However, a good controller should also be a robust controller. For this case, we developed a controller which could handle carts of mass 2 to 20 kilograms. This controller has the same structure as the 555 controller and will be referred to as the 555** controller. To develop the 555** controller, a slight modification was made to the fitness function. Essentially, when fitness was evaluated, a simulation was made at the two mass extremes for each initial condition, and the fitness was the sum of the simulations at each weight. Also, the upper and lower limits on the force was reduced as a compromise between the two weight extremes. The new limits were $\pm 75N$

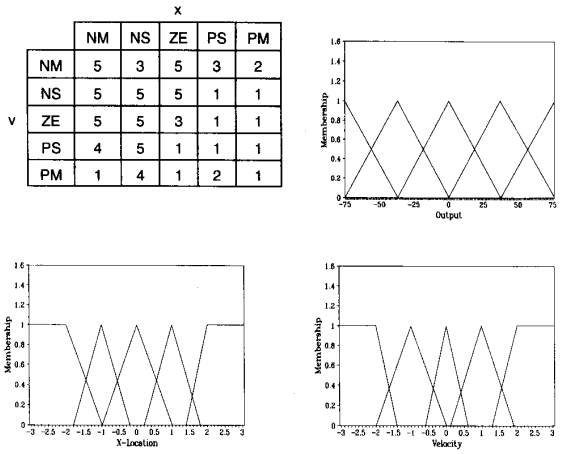

Fig. 7. Robust 555** fuzzy cart controller.

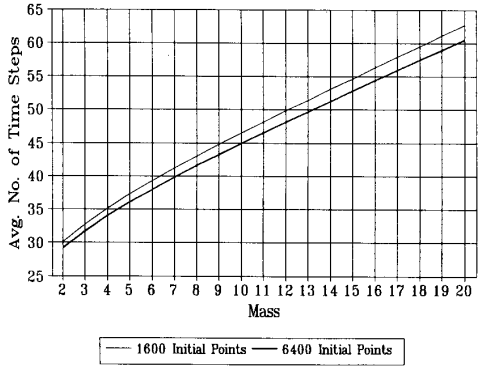

Fig. 8. Performance of Robust 555** Fuzzy Cart Controller.

instead of the previous $\pm 150N$. The 555** controller designed by a GA is shown in Fig. 7.

To test the robust controller, a simulation was made from masses ranging from 2 kg to 20 kg in 1kg increments. As was done before, the input spaces from -2 to $+2$ were divided in increments of 0.1 m for x-location and 0.1 m/s for velocity, yielding 1600 initial conditions for each mass. Also, another run was made with the input spaces divided into 80 points (0.05 increments). This gave 6400 initial conditions tested for each cart mass. The average number of time steps for each mass is shown in Fig. 8. The controller had no failures for any of the initial conditions examined, showing its ability to deal with changing masses. As the figure also shows, the average number of time steps needed actually decreased when more initial conditions were added, lending confidence to its ability to handle any possible input combinations.

F. Influence of Faulty Rules

While the 555 controller behaved well for the given ranges for the input variables, it was desired to know how the controller would perform if any one of the rules became faulty. Would the system diverge because of the bad rule or would it simply slow the controller down? Table V shows the results

TABLE V
EXAMINATION OF RULE CHANGES FOR 555 FUZZY CART CONTROLLER

location (row, column)				
(1,1)	5 to 1	5 to 2	5 to 3	5 to
divergence errors	30	21	5	0
time errors	0	0	0	0
(2,2)	5 to 1	5 to 2	5 to 3	5 to
divergence errors	0	0	0	0
time errors	0	0	0	0
(3,3)	3 to 1	3 to 2	3 to 4	3 to
divergence errors	0	0	0	0
time errors	22	0	0	22

TABLE VI
INITIAL CONDITIONS FOR TRUCK-BACKING SYSTEM

Controller	Initial Conditions (x_0, ϕ_0)
757	(10,-45) (10,22.5) (10,90) (10,157.5) (10,225) (40,22.5) (40,157.5) (50,-45) (50,90) (50,225) (60,22.5) (60,157.5) (90,-45)

from changing some of the rules in the 555 controller. For example, the rule at location (1, 1) in the rule set matrix was initially 5 (indicating a positive medium output). This rule was changed to 1, 2, 3, and 4 while noting the performance on each of the 1600 initial points.

Table V makes many interesting points. In many instances, the rules surrounding the bad rule can, in effect, overrule it when the calculation of the output is determined. Even though this occurs, there will be some degradation in response time because a rule that is the complete opposite of the surrounding rules will have a great influence on the output. This effect was seen in the change of rule (3, 3), which is the steady-state rule, IF (x = zero) and (v = zero) THEN (force = zero). When the output was at the two extremes, 1 and 5, the controller was not able to bring the final values within the tolerance before "timing out". Rules along the edges and the corners such as (1, 1), however, did not have the luxury of having rules completely surround them and able to compensate for their mistakes. As can be seen from modifying (1, 1), the further the rule was from its original value, the more likely the chance of divergence. Inner rules, such as (2, 2) did not suffer from this problem.

IX. RESULTS FOR TRUCK-BACKING PROBLEM

A. Initial Conditions

The controller used for this problem was a 757 controller. These values were chosen so a comparison could be made with the results illustrated by Kosko [6]. The initial conditions used are shown in Table VI. These initial conditions include four points added intentionally to prevent the wrapping around effect of ϕ mentioned earlier. Those four points are at (10, 22.5 degrees), (10, 157.5 degrees), (90, 22.5 degrees), and (90, 157.5 degrees). Without these extra points, the GA might try to wrap around to get to (x = 50 m, y = 100 m, ϕ = 90 degrees) instead of taking the shorter, more direct route.

B. Fitness Function

As was done for the cart controller, the fitness function was divided into two stages, the first stage for development of working controllers and the second stage for time minimization. As before, the fitness function was of a reinforcement/reward type. To promote better results from the GA, the tolerance variables, x_{tol} and ϕ_{tol}, used to determine if a simulation was successful, were allowed to decrease linearly from generation 0 (where their respective values were ± 7.5 m and ± 18.0 degrees) to generation 30 (where their respective values were ± 0.5 m and ± 1.3 degrees). Given that x ranges over 100 m and ϕ ranges over 360 degrees, the final tolerance values represent errors of only 1% and 0.722%. These tolerance values were compared to error values, x_{err} and ϕ_{err}, to determine if a simulation should end. The values for x_{err} and ϕ_{err} are given as

$$x_{\text{err}} = 50.0 - x$$

and

$$\phi_{\text{err}} = 90.0^{\circ} - \phi.$$

The fitness for the first 30 generations is shown in Appendix A. As was the case for the cart controller, the fitness function was determined experimentally; however, the desire to keep the fitness function simple led to a much more manageable equation.

If the controller did not meet the tolerance requirements or it "timed out," it was punished with a negative fitness. In the early stages, however, controllers that did meet the tolerance requirements are highly valued, and are thus rewarded with a fitness that would offset four failures.

For the second stage, the fitness function is totally time dependent. The second fitness function, used from generation 31 to generation 100 is given in Appendix A.

Here x_{tol} and ϕ_{tol} maintained the values they obtained at generation 30 (± 0.5 m and ± 1.3 degrees, respectively). Since the average time the controller took for a given initial condition was approximately 50 time steps, one successful simulation was approximately equal to one failed simulation.

C. 757 Controller

The string length for the 757 controller problem was 54 (35 for the rule set, seven for the ϕ membership function, five for the x membership function, and seven for the θ membership function). The string resembles the example shown in Fig. 1 except for the change in variable names and the additional alleles needed for the increased rules, added fuzzy sets, and the inclusion of the output variable. The controller developed by the GA is given in Fig. 9.

To see physically how the truck controller performed, a program was written which would track the motion of the truck from a given beginning point and angle. An example of the truck controller's behavior is given in Fig. 10.

D. Comparison with Reference Controller

Now that a GA had designed the entire controller, the true test came in its comparison with another controller, the one

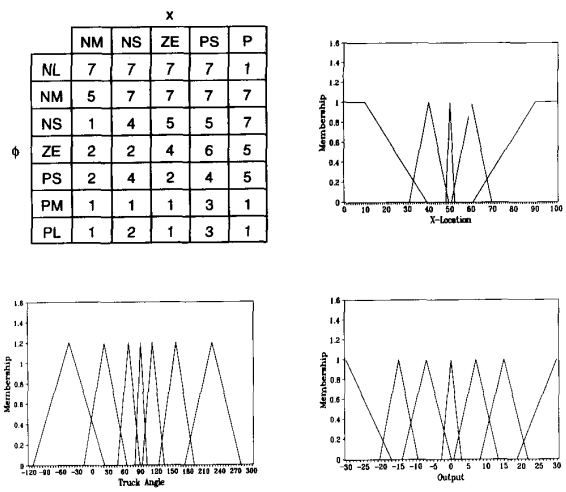

Fig. 9. 757 fuzzy truck controller.

used by Kosko. Kosko had compared his fuzzy controller with a neural controller, and his results indicated the fuzzy controller consistently outperformed it. The basis of judging the performance was the same as for the cart controller, the average number of time steps required when tested on a given set of initial conditions. For this purpose, the x input space was divided in increments of 1 from 5 to 95. The ϕ input space was divided in increments of 1 degree from -90 degrees to 270 degrees. This means the total number of initial conditions used was 32,400 (90×360). The results of the GA-designed controller and the reference controller are shown in Table VII. Dividing the number of time steps for the reference controller by the number of time steps for the GA-designed controller, we find that the reference controller averaged 6.34% longer, showing the GA made a small, but relevant improvement. As before, examining the input spaces in this manner does not ensure the stability of the controller, but given the large number of points considered, stability is almost certain.

X. Conclusion

This paper clearly shows the potential for using genetic algorithms to solve optimization problems. The ability of fuzzy logic controllers to provide control where more conventional methods become too complex has also been shown by researchers. This work has shown these two, fairly new, methods can be used together to form controllers without the previously needed human expert. This methodology allows the complete design of both major components of fuzzy controllers, the rule sets and membership functions, leading to high performing controllers which are completely computer-designed. We have developed four different controllers for the cart problem, each of which was able to bring the cart to equilibrium over the entire ranges of the input spaces. Also, we have shown a GA's ability to design a robust controller which can work over a wide parameter range. The GA designed controllers proved to show the same degree of fault tolerance one would expect in

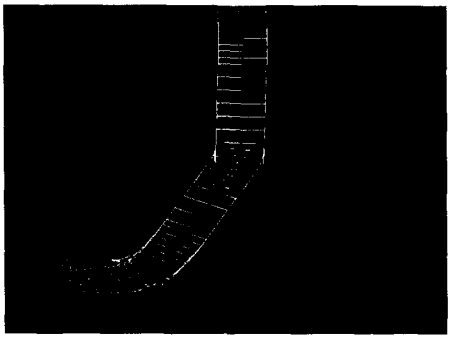

Fig. 10. Motion of truck from $x_0 = 20$, $\phi_0 = -80$ degrees.

TABLE VII
COMPARISON OF GA-DESIGNED AND REFERENCE CONTROLLERS

	GA- Designed	Expert Designed
time steps required	1,574,255	1,674,124
average no. of time	48.59	51.67
failures	0	0

a fuzzy controller. In addition, we have developed a truck-backing controller which outperformed a reference controller. While these results are encouraging, more work should be done on refining the process. First, fitness functions are currently a weak point; better methods need to be developed to find the best fitness functions. Also, more work needs to be done to ensure that the GA finds controllers which operate over the entire space of the input variables. Also, as mentioned earlier, the inclusion of finding the location of the peaks of the triangles in the membership functions will yield even higher performing controllers. Finally, controllers for still more problems should be examined to show the effectiveness of this method.

APPENDIX A

Fitness Functions for Cart and Truck Controllers Cart Controller

Evolution Stage Fitness Function

if ($|x| < 0.5$) and ($|\text{velocity}| < 0.5$) then { x and v are within tolerance values }
fitness = 8 * 175/time
else if (time = 175) then
{ simulation times out before x and v
if ($|x| < 1.0$) and ($|\text{velocity}| < 1.0$) then are within tolerance values}
fitness = 3.5/sqrt(x^2 + velocity^2)
else
fitness = -1
else { $x > 5.0$ or $v > 5.0$}
fitness = -7

Refinement Stage Fitness Function

if ($|x| < 0.5$) and ($|\text{velocity}| < 0.5$) then { x and v are within tolerance values }
fitness = 0.3 * (175 − time)
else if (time = 175) then
{ simulation times out before x and v
fitness = -42 * sqrt(x^2 + velocity^2)
are within tolerance values}
else { $x > 5.0$ or $v > 5.0$}
fitness = -300

Truck Controller

Evolution Stage Fitness Function

if ($|x_{\text{err}} < x_{\text{tol}}$) and ($|\phi_{\text{err}}| < \phi_{\text{tol}}$) then
fitness = 2.0
else
fitness = -0.5

Refinement Stage Fitness Function

if ($|x_{\text{err}}| < x_{\text{tol}}$) and ($|\phi_{\text{err}}| < \phi_{\text{tol}}$) then
fitness = 100 − time
else
fitness = -50

REFERENCES

[1] L. A. Zadeh, "Fuzzy sets," *Informa. Contr.*, vol. 8, pp. 338–353, 1965.
[2] C. L. Karr and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 1, pp. 46–53, Feb. 1993.
[3] C. L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," in *Proc. the Fourth Int. Conf. Genetic Algorithms*, 1991, pp. 450–457.
[4] P. Thrift, "Fuzzy logic synthesis with genetic algorithms," in *Proc. Fourth Int. Conf. Genetic Algorithms*, pp. 509–513, 1991.
[5] D. Nguyen and B. Widrow, "The truck backer-upper: An example of self-learning in neural networks," in *Proc. Int. Joint Conf. Neural Networks (IJCNN-90)*, vol. 2, 1989, pp. 357–363.
[6] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1992.
[7] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan, 1975.
[8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[9] R. Jain, "Fuzzyism and Real World Problems," in *Fuzzy Sets: Theory and Applications to Policy Analysis and Information Systems*, Wang, P. P, and Chang, S. K., Eds. New York: Plenum, 1980.
[10] Y. Leung, *Spatial Analysis and Planning Under Imprecision*. New York, NY: Elsevier, 1988.
[11] K. Krishnakumar, "Microgenetic algorithms for stationary and nonstationary function optimization," in *SPIE Proc. Intell. Contr. Adaptive Syst.*, vol. 1196, 1989, pp. 289–296.
[12] T. J. Procyk and E. H. Mamdani, "A linguistic self-organizing process controller," *Automatica*, vol. 15, no. 1, pp. 15–30, 1979.
[13] A. Homaifar and V. E. McCormick, "Full design of fuzzy controllers using genetic algorithms," in *SPIE Conf. Neural Stochastic Methods Image Signal Process.*, vol. 1766, 1992, pp. 393–404.
[14] H. Nomura, I. Hayashi, and N. Wakami, "A self-tuning method of fuzzy reasoning by genetic algorithm," in *Proc. Int. Fuzzy Syst. Intell. Contr. Conf. (IFSICC '92)*, 1992, pp. 236–245.
[15] M. A. Lee and H. Takagi, "Integrating design stages of fuzzy systems using genetic algorithms," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE '93)*, 1993, pp. 612–617.
[16] T. Nishiyama, T. Takagi, R. Yager, and S. Nakanishi, "Automatic generation of fuzzy inference rules by genetic algorithms," in *Proc. 8th Fuzzy Syst. Symp.*, 1992, pp. 237–240. (in Japanese)
[17] Y. Qian, P. Tessier, and G. Dumont, "Fuzzy logic based modeling and optimization," in *2nd Int. Conf. Fuzzy Logic and Neural Networks (IIZUKA '92)*, 1992, pp. 349–352.
[18] T. Tsuchiya, Y. Matsubara, and M. Nagamachi, "Learning fuzzy rule parameters using genetic algorithms," in *Proc. 8th Fuzzy Syst. Symp.*, 1992, pp. 245–248. (in Japanese)
[19] L. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst. Man, Cybern.*, vol. 22, no. 6, pp. 1414–1427, Nov.-Dec. 1992.

Abdollah Homaifar received the B.S. and M.S. degrees from State University of New York at Stony Brook in 1979, and 1980, respectively, and the Ph.D. degree from the University of Alabama in 1987, all in electrical engineering.

He is currently an Assistant Professor in the Department of Electrical Engineering at the North Carolina A&T State University. His current research interests include the application of genetic algorithms and fuzzy logic to the design of a controller for a high speed civil transport vehicle as well as machine learning, expert systems, adaptive control, optimal control, signal processing, and fuzzy control and modeling.

Dr. Homaifar is an associate editor of the Journal of Intelligent Automation and Soft Computing. He is a member of IEEE Control Systems Society, Sigma Xi, Tau Beta Pi, and Eta Kapa Nu.

Ed McCormick (M'94) received the B.S. degree in aerospace engineering in 1987 from North Carolina State University in Raleigh, N.C. He received the M.S. degree in electrical engineering in 1992 from North Carolina Agricultural and Technical State University in Greensboro, N.C.

Through his advisor at North Carolina A&T, he gained an interest in genetic algorithms and fuzzy logic. Since 1992, he has been with Research Triangle Institute in Research Triangle Park, N.C. He is now working with the Environmental Protection Agency to increase AC induction motor efficiency with fuzzy logic. His current research interests include fuzzy systems, genetic algorithms, and linear and nonlinear control systems.