On Genetic Programming of Fuzzy Rule-Based Systems for Intelligent Control

Edward Tunstel* and Mo Jamshidi
NASA Center for Autonomous Control Engineering
Department of Electrical and Computer Engineering
University of New Mexico
Albuquerque, NM 87131 USA

International Journal of Intelligent Automation & Soft Computing, Vol. 2 No. 3, pp. 273–284, 1996

Abstract

Fuzzy logic and evolutionary computation have proven to be convenient tools for handling real-world uncertainty and designing control systems, respectively. An approach is presented that combines attributes of these paradigms for the purpose of developing intelligent control systems. The potential of the genetic programming paradigm (GP) for learning rules for use in fuzzy logic controllers (FLCs) is evaluated by focussing on the problem of discovering a controller for mobile robot path tracking. Performance results of incomplete rule-bases compare favorably to those of a complete FLC designed by the usual trial-and-error approach. A constrained syntactic representation supported by structure-preserving genetic operators is also introduced.

Keywords: fuzzy control, genetic programming, syntactic contraints, mobile robots, rule-base discovery.

1 Introduction

Recent research and applications employing non-analytical methods of soft computing such as fuzzy logic and evolutionary computation has demonstrated the utility and potential of these paradigms for developing intelligent control systems [1, 2]. Fuzzy logic control and evolutionary computation have proven to be convenient tools for handling real-world uncertainty and knowledge representation, and the design of intelligent control systems, respectively [3, 4]. Here, we present an approach that exploits the combined attributes of these paradigms for the purpose of developing intelligent algorithms for controlling autonomous dynamic systems that interact with the real world. In particular, we apply the genetic programming paradigm (GP) [2] to the problem of learning rules for use in a fuzzy rule-based control system. In genetic programming, a population is comprised of computer programs (individuals) that are candidate solutions to a particular problem. The programs are structured as hierarchical compositions of functions and terminals (arguments) of various sizes and shapes. These individuals participate in a simulated evolutionary process wherein the population evolves over time in response to selective pressure induced by the relative fitnesses of the individuals in a particular problem environment. In this work, each individual is coded as a LISP symbolic expression (S-expression) that implements condition-action statements which collectively serve as a rule-base to be embedded in a fuzzy-logic controller. The GP concept can be implemented in other programming languages as well, at both high and low levels [5, 6].

*On leave from NASA/Jet Propulsion Laboratory, Pasadena, CA.

Our approach to this problem calls for the use of a constrained syntactic structure to construct each individual S-expression in the population. This required the definition of syntactic rules of construction and structure-preserving GP operators for breeding the resulting individuals.

It has been observed that the artificial evolution of computer programs may produce deterministic control strategies that have slightly different features than those produced by humans. The existence of similar building blocks in the strategies of artificial evolution and those of humans has also been observed [7]. These things considered, the following questions come to mind. What innovations in control strategies, if any, can we expect from artificial evolution for a given control problem for which a human-derived solution exists? What is the potential of genetic programming for evolution of fuzzy controllers? In this article, we present the results of our attempt to provide an answer to the latter question, as its answer is related to that of the former. We do this by focussing on a particular control problem, namely, determining a fuzzy linguistic rule-base for steering a mobile robot onto a desired path. We use genetic programming to evolve the rule-base and compare its performance with a fuzzy controller produced by the authors via a trial-and-error design approach.

Before describing the control problem, we give a brief overview of fuzzy logic control. This is followed by Section 3 where motivations are given for seeking automatic design methods for fuzzy controllers, and why GP was considered as a possible means to that end. In Section 4 we acknowledge work related to the research reported herein. The remainder of the article covers details of our GP implementation, results of the steering control application, conclusions, and improvements to the approach.

2 Overview of Fuzzy Logic Control

Fuzzy control is one of the more active areas of application of fuzzy logic and the underlying fuzzy set theory introduced by Zadeh [8]. Fuzzy logic controllers are intelligent control systems that smoothly interpolate between rules, i.e. rules fire to continuous degrees and the multiple resultant actions are combined into an interpolated result. Processing of uncertain information and saving of energy using common-sense rules and linguistic statements are the basis for fuzzy logic control. Fuzzy sets may be represented by a mathematical formulation often known as the membership function. This function gives a degree or grade of membership within a fuzzy set. Over a given universe of discourse X, the membership function of a fuzzy set A, denoted by $\mu_A(x)$, maps the elements $x \in X$ into a numerical value in the unit interval, i.e.

$$\mu_A(x) : X \rightarrow [0, 1]. \tag{1}$$

Within this framework, a membership value of zero corresponds to an element which is definitely not a member of the fuzzy set, while a value of one corresponds to the case where an element is definitely a member of the set[1].

Implementation of a fuzzy controller requires assigning membership functions for both inputs and outputs, thus the membership values are actually measures of degree of causality in an input-output mapping. Inputs to a fuzzy controller are usually measured variables, associated with the state of the controlled plant, that are fuzzified (assigned membership values) before being processed by an inference engine. The heart of the controller inference engine is a set of IF-THEN rules whose antecedents and consequences are made up of linguistic variables and associated fuzzy membership functions. Consequences from different rules are numerically aggregated by fuzzy disjunction (typically union via the *max* operator) and are then collapsed (defuzzified) to yield a single real number output that serves as the control signal for the plant. For a more detailed introduction to fuzzy control, fuzzy set operations, and the concepts of fuzzification, inference, aggregation, and defuzzification see one of [1, 8].

3 Motivation

The establishment of a systematic approach to the design of fuzzy controllers in the absence of an expert, or sufficient knowledge of the problem domain, is currently an open problem. The approach often taken is an iterative one of trial-and-error. It typically involves tweaking of membership functions used to express the uncertainty in inputs and outputs, as well as modifications of the fuzzy rule-base. This process, which leads to a fuzzy controller that performs well according to the designer's subjective evaluation, can turn out to be quite lengthy depending on the complexity of the control problem. This serves to weaken the scalability of one of the strongest attributes of fuzzy logic applications in control — fast development time. Various attempts have been made to address this design issue. These include the determination of fuzzy membership functions and rules by optimization or search using genetic algorithms (GAs), and by learning using neural networks [4] [9]–[11]. When using these techniques to determine rule sets for fuzzy systems, it is often necessary to encode the rules in a numerical form suitable for processing by the GA or the NN, and subsequently decode them into the appropriate linguistic terminology. For the GA, the chromosome representing a rule is typically a string of numerical genes whose alleles belong to binary or n-ary alphabets, and/or the set of real or natural numbers, rather than the collection of linguistic variables, fuzzy sets and fuzzy logic connectives that actually make up the rule. An exception is the representation proposed in [12] where a chromosome is encoded as a matrix whose elements (alleles) are fuzzy sets. Furthermore, for approaches that use the simple GA [13], the fixed-length chromosome restricts each individual to have the same pre-specified number of rules. The contention here is that the genetic programming paradigm may offer a more direct approach to fuzzy controller design.

The main difference between the GP paradigm and that of the simple GA is the increase in complexity of the structures undergoing adaptation, i.e. parse trees of hierarchical computer programs rather than numeric strings [2]. In our application of GP to evolution of fuzzy rule-based systems, the same fuzzy linguistic terms and operators that comprise the genes and chromosome persist in the phenotype. Thus, the use of GP eliminates the need for encoding/decoding of the fuzzy linguistic rule set. Furthermore, our implementation allows for a population of individuals of various sizes and different numbers of rules. In this article, no claims are made about the relative performance of GAs versus GP as tools for search, optimization, or learning. After all, "GP is a GA where critical choices have been made to suit its goal of program discovery" [14]. The advocacy of GP for evolving fuzzy rule-bases is rooted in its convenience of representation as it pertains to fuzzy system design. In our view GP seems to be more appropriate for design of fuzzy rule-bases since it can facilitate the manipulation of linguistic variables directly associated with the problem.

4 Related Research

The artificial evolution of rules for systems control has been investigated by a number of researchers. Most have focussed on using models of Darwinian evolution. In recent years the body of related literature has become quite extensive. It is not the intention in this article to provide an inclusive overview, but rather to acknowledge prior and ongoing research that most closely relates to the approach described herein. Interested readers may consult a recently compiled bibliography [15] for a broader overview. In [16], Goldberg demonstrated the effectiveness of learning classifier systems at learning rules to control position of a simple inertial object and to control a simulated natural gas pipeline. Grefenstette and Schultz have developed the SAMUEL system for learning control rules [17]–[19]. The system has proved successful at robot control problems as well as simulated control of evasive maneuvers for tactical aircraft. They introduced a restricted high-level rule language (and associated genetic operators) that distinguishes their GA approach from others that are based on the string representation of chromosomes. The resulting

approach has strong similarities to the work described here. The fundamental differences lie in our use of GP instead of GA, fuzzy sets instead of crisp sets, and linguistic variables rather than numeric variables. Harvey et al [21] have concentrated on evolving robot behavior using GAs in conjunction with neural networks. They suggest that building controllers by hand becomes prohibitively difficult for increasingly complex behavior. This view is shared by Feldman [11] who has developed a technique that encodes fuzzy control rules as a fuzzy network, a connectionist extension to fuzzy linguistic systems. The GA is used to synthesize or modify the rules of the fuzzy network controller. Finally, Kinzel et al [12] deemed it necessary to modify the GA (using the matrix rule-base representation mentioned earlier) by taking the properties of fuzzy controllers into account to facilitate fast convergence. As a departure from the Darwinian approach Grefenstette [20] added Lamarckian mechanisms to the SAMUEL system that improve the quality and computational cost of rule learning for control.

Koza [2] has applied genetic programming to a number of control problems related to our interests, namely, the truck backer-upper problem and the evolution of robot subsumption behaviors for wall-following and box-pushing. Shortly after the publication of Koza's text, applications of genetic programming to control problems of the type we focus on here have appeared in the literature. The most notable relation to this work is that of Reynolds [22]. He has used GP to evolve corridor following behaviors for a simulated robot in the presence of noise. Similar work has been done by Fraser [23] in evolving multi-agent emergent behaviors, and Handley [24] in mobile robot path planning.

We have already mentioned the difference between our approach and that of SAMUEL. With the exception of that system and the work on fuzzy controller evolution, all of these applications of evolutionary computation result in evolved controllers that are deterministic computer programs based on binary logic reasoning. Each of the GA implementations that use string representations for chromosomes employ the binary encoding scheme. Each of the GP implementations make use of numeric values as terminals. Thus, the work described herein differs from the related work in either its focus on the evolution of *fuzzy* systems based on approximate reasoning, its use of GP with *linguistic* terminals and fuzzy logic operators as functions, or both.

5 Genetic Programming Implementation

In the genetic programming paradigm, the program search space is contained in the set of all possible S-expressions that can be composed recursively from a set of *functions* and a set of *terminals*. Each function in the function set, F, takes a specified number of arguments. In general, functions may include arithmetic operators, mathematical functions (e.g. sine, cosine, absolute value), Boolean operators, conditionals, etc. Terminals in the terminal set T are typically either variables or constant atoms. For the sake of brevity, let us introduce the control problem addressed here before describing the details of the GP implementation. This will allow us to discuss the implementation issues within the context of the problem.

5.1 Mobile robot steering control problem

The problem is to find a fuzzy rule-base to properly steer a mobile robot for path following in the plane. The problem is taken from Hemami [25] where it is formulated for a class of low-speed (less than 2 m/s) tricycle-model vehicles. Hemami derived a state-space model, based on the robot kinematics, where the state vector consisted of measurable position (ε_d) and orientation (ε_θ) errors associated with path following (see Figure 1). The steering angle (δ) is the corrective control action that causes the error states to decay to zero, thus forcing the robot to follow the path. The position error is taken as the deviation from the nearest point on the desired path. The orientation error is the angular deviation

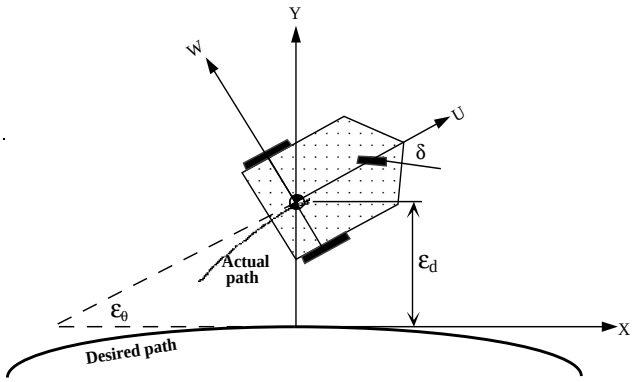

Figure 1: Control and error variables associated with a desired path.

of the robot's heading from the tangent to the desired path. The rule-base we wish to evolve is for a two-input-one-output fuzzy controller that will map the error states into a steering angle at each time step during the robot's attempt to follow a desired path. This is a fundamental motion capability that is often an integral part of more complex behavioral repertoires for autonomous mobile robots [26]. Based on the geometry of the problem as formulated in [25] the errors fall into eight different categories that are pair-wise symmetric. Four of these are:

$$(a)\ \varepsilon_d = 0, \varepsilon_\theta < 0;\ (b)\ \varepsilon_d < 0, \varepsilon_\theta = 0;\ (c)\ \varepsilon_d < 0, \varepsilon_\theta < 0;\ (d)\ \varepsilon_d > 0, \varepsilon_\theta < 0 \tag{2}$$

the remaining categories are symmetric to these.

In the current study we focus our efforts on evolving the rule-base and assume that the membership functions are specified *a priori* and are fixed. The membership functions are taken from an existing fuzzy control solution hand-derived and refined through trial-and-error by the authors. Five fuzzy sets are used for each of the input and output linguistic variables. Practical universes of discourse were defined such that $|\varepsilon_d| < 1$m, $|\varepsilon_\theta| < \frac{\pi}{3}$rad, and $|\delta| < \frac{\pi}{4}$rad. The hand-derived rule-base consists of R_c=25 rules, which is *complete*, i.e. there is a rule for all combinations of input fuzzy sets taken two at a time. The set of GP alleles, $F \cup T$, includes functions that perform fuzzy set/logic operations and terminals that are inputs, outputs, and their corresponding membership functions. These are described below.

5.2 Genetic Programming Functions and Terminals

For the purpose of evolving fuzzy rule-bases (which are essentially programs of fuzzy conditional statements) we have chosen the following function set,

$$F = \{\texttt{ANT}, \texttt{CONSQ}, \texttt{f\_AND}, \texttt{IF}-\texttt{THEN}, \texttt{f\_OR}\} \tag{3}$$

with `f_OR` (described below) taking a variable number of arguments (equal to the number of rules) and the remaining functions each taking two arguments. The function `ANT` represents a fuzzy proposition in the antecedent of a fuzzy rule. Its arguments are an input linguistic variable and an associated fuzzy membership function. For example, in the proposition, *error is LARGE*, *error* is a linguisitic variable and *LARGE* designates a membership function expressing the "meaning" of the current value of *error*. `ANT` returns a numerical value in the closed interval [0, 1] representing the membership value, or degree of truth, of the proposition. Note that if a rule contains only one proposition in its antecedent the membership value represents the rule strength. `CONSQ` is defined in a similar manner for output linguistic variables and fuzzy sets except that it returns the output fuzzy set designated in the rule consequence. The

f_AND function is simply the fuzzy intersection operator of fuzzy set theory. It performs the conjunction of two or more fuzzy propositions yielding a numerical value for the rule strength. The f_AND function can be defined using any t-norm; *min* and *product* are most commonly used in fuzzy control [1]. Here we limit it to the conjunction of two propositions with the idea that conjunctive forms of higher order can be constructed by recursive calls to the function (the level of recursion is bound by a specified maximum depth of the rule tree). In addition, the current implementation restricts f_AND to occur only in rule antecedents. Therefore, its two arguments can be return-values of either ANT or a recursive call to itself. The function representing a rule is IF-THEN. Its first argument is the rule strength returned by either ANT or f_AND; its second argument is the fuzzy set returned by CONSQ. Finally, the function f_OR serves as a fuzzy aggregation operator. It occupies the root node of every tree in the population of rule-bases. Each rule that fires in a fuzzy rule-base returns a fuzzy set as the result of the rule consequence. f_OR operates on the output fuzzy sets by taking their fuzzy union to produce a resultant fuzzy set representing the overall output of the rule-base. This overall output fuzzy set is the return-value of an individual rule-base in the population. Consequently, a wrapper (output interface) for S-expressions is the center-of-gravity defuzzification operator which defuzzifies the fuzzy output to yield a real number for the control signal.

The terminal set is made up of the input and output linguistic variables and the corresponding fuzzy sets associated with the problem being solved. For the steering controller the terminal set is defined as

$$T = \{\varepsilon_d, \varepsilon_\theta, \delta, pNB, pNS, pZ, pPS, pPB, oNB, oNS, oZ, oPS, oPB, NB, NS, Z, PS, PB\} \qquad (4)$$

where the linguistic notation is as follows: $NB \equiv$ "negative big", $NS \equiv$ "negative small", $Z \equiv$ "zero", $PS \equiv$ "positive small", $PB \equiv$ "positive big". The lowercase prefixes "p" and "o" designate fuzzy sets for position error and orientation error respectively. Fuzzy sets for the steering angle are labeled without a prefix. The elements of the function and terminal sets comprise the linguistic terminology of the problem. Therefore, there is no need to convert between numeric and symbolic representations when using genetic programming to evolve a fuzzy rule-base.

5.3 Syntactic Constraints and Structure-preserving Operators

In many genetic programming applications, unrestricted S-expressions are sufficient to solve a problem given a function set and a terminal set that satisfies the closure property [2]. That is, each function in F should be well defined and closed for any combination of arguments that it may encounter. As a result, individuals may have any composition of elements from the combined set, $F \cup T$, occupy the nodes of the tree with the only restriction being that the root must be a function and the leaves must be terminals. This is not the case here. Our GP implementation imposes strong constraints on the syntax of a rule-base that are defined by special rules of construction.

A rule-base that could potentially evolve from the designated function set and terminal set can be expressed as a rooted, point-labeled tree with preordered branches. An example of a syntactically valid rule-base of two rules and a depth of five is depicted in Figure 2 along with its interpretation as a linguistic rule-base. From the figure one can imagine how arbitrary placement of functions and terminals in this tree could lead to severe syntactic violations. Valid rule-bases must conform to the following syntactic rules of construction:

- f_OR must occupy the root of the tree and cannot occur at non-root points.
- Only IF-THEN is allowed at the level immediately below the root.
- A left-child of IF-THEN can only be ANT or f_AND.
- A right-child of IF-THEN can only be CONSQ.
- A child of f_AND can be either ANT or f_AND.
- A child of ANT can only be input linguistic variables and input fuzzy sets.

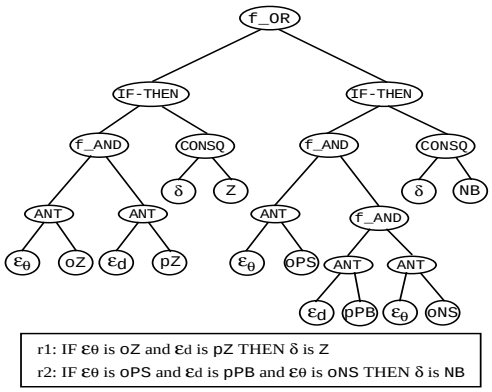

Figure 2: Rule-base tree satisfying syntactic constraints.

- A child of `CONSQ` can only be output linguistic variables and output fuzzy sets.

Additional ramifications of these syntactic constraints are that full trees are not possible if the number of inputs and outputs is not equal, and extra care must be taken to ensure that linguistic variables are paired with appropriate fuzzy sets as children of `ANT` and `CONSQ` nodes. The minimum depth of a valid rule tree is three; this corresponds to rules with a single antecedent. The maximum number of antecedents per rule is 2^{d-3}, where d is the maximum permissible depth of the rule tree specified as a control parameter of the GP run.

All rule-bases in the initial population are randomly created using these rules, but descendant populations are created by the reproduction, crossover, and mutation operators. The offspring of rule-bases modified by crossover and mutation must also conform to the syntactic structure. Structure-preserving crossover is achieved by randomly selecting any non-root node as the crossover point in the first parent, and restricting the crossover point in the second parent to be a randomly-selected point of the same type. One exception is that `ANT` and `f_AND` make a valid pair of crossover points provided that one of the resulting offsprings do not violate the preset maximum depth for rule-base trees. The crossover is completed in the usual way [2] by swapping the subtrees at (and including) the crossover points of the two parents. This crossover operator not only preserves the syntactic structure of the rule-base but it also preserves the context of subtrees, particularly when function nodes are selected as crossover points. This issue of context preservation in GP has been recently addressed by D'haeseleer [27], where he introduces two new crossover operators that provide a more flexible mechanism to decouple the evolution of different branches of an individual tree. In our implementation, context preservation is a necessary by-product of the syntactic constraints imposed by the rule-base structure. Structure-preserving mutation is done by randomly selecting a non-root point in a rule-base tree, discarding the selected point and the nodes below it, and replacing the discarded portion with a randomly-generated (but syntactically valid) subtree at that point. The root node is protected from both crossover and mutation.

Amidst all of the constraints on syntax and structure of the fuzzy rule-bases, there is room for some flexibility. In the creation of the initial population, the number of rules (number of arguments to `f_OR`) in each rule-base is assigned to be a random integer in the interval, $[R_{min}, R_{max}]$, specified before the run. The value for R_{min} is chosen as a lower bound on the size of a rule-base that the control engineer feels may be sufficient to control the system. The upper bound can be chosen such that $R_{max} \geq R_c$, the number of rules required for a complete rule-base. In this study we use [10, 30]. This feature of the implementation is important for ensuring diversity in the population as it allows for rule-bases of different sizes. It also increases the potential for finding a rule-base of smaller size than necessary for completeness. It is well-known to practitioners of fuzzy control that a number of dynamic systems exist

that can be controlled using fewer rules than dictated by the value of R_c for the fuzzy rule-base. Finally, it should be noted that some unusual circumstances regarding allowable rules result from the imposed structure. It is possible, for example, for an input linguistic variable to appear more than once in the antecedent of a rule (examine Figure 2). In fact, it is also possible for a given fuzzy proposition to have multiple occurrences in a rule (in this case redundant occurrences are deleted if they persist in the final solution). While one could argue that this unnecessarily enlarges the search space, we allow such unusual possibilities to prevent restricting GP from discovering innovative control strategies that may be counter-intuitive to the human designer, and consequently overlooked.

6 Results and Discussion

In these initial investigations the GP system was run using small population sizes of 10–20 rule-bases for a number of generations ranging from 9–46. In GP, genetic diversity remains high even for very small populations due to the tree structure of individuals [2]. Twelve GP runs were executed for the steering control problem described above. In this Section, results from several representative runs in which the best-of-run rule-base performed well with respect to the hand-derived rule-base are presented.

The simulated robot is based on Hemami's kinematic model and approximate dimensions are taken from the Hero-1 mobile robot — a 0.3m wheelbase, and a rear-axle-to-front-wheel offset of 0.2 m. In all of the simulations the robot travels at a constant speed of 1.5 m/s. Following Hemami's formulation, it is assumed that the error states are measurable. Thus, the robot has access to the error states (perhaps via dead-reckoning sensors) at all times. It is also assumed here that the source of sensory information has practical uncertainties and imprecision associated with it. The simulations are conducted at 20 Hz (i.e. time steps of 0.05 seconds) for a maximum of 5 seconds. We use eight fitness cases corresponding to initial conditions selected from each of the eight error categories mentioned in Section 5.1. This is a small number relative to the number of fitness cases typically used for problems solved to date using GP. Ideally, the number should be chosen such that the fitness cases represent the search space sufficiently to allow the evolved control strategy to generalize (i.e. handle unforeseen initial conditions). Given the pairwise-symmetric error categories for the steering control problem, eight is a convenient choice. Moreover, in mobile robot problems involving time-consuming simulation of each fitness case per individual, small numbers of fitness cases and/or small population sizes are often necessary tradeoffs. During the GP evolution process, each rule-base in the current population is evaluated to determine its fitness for steering the robot onto the desired path. This evaluation is achieved by simulating the robot's motion from each of the eight initial conditions until either the goal state is reached or time expires. The raw fitness of a rule-base is defined as the sum, over the fitness cases, of the Euclidian norms of the error state vector at the end of each fitness case, i.e.

$$Raw\,fitness = \sum_{i=1}^{8} \sqrt{(\varepsilon_d^2 + \varepsilon_\theta^2)_i} \tag{5}$$

Among the measures of fitness used in [2], standardized fitness (i.e. lowest numerical values imply best fit) is predominant particularly in problems for which the objective is to minimize costs such as error. In this problem standardized fitness is equivalent to raw fitness. Thus, a perfect score is zero and lower raw fitnesses are associated with better rule-bases. In addition to scoring the best fitness, we would like candidate rule-bases to cause the error states to decay to final values within specified tolerances ($|\varepsilon_d| <$ 0.15m and $|\varepsilon_\theta| < 0.26$ rad.) for each fitness case. We refer to such an event as a *hit*, the conditions for which are used as metrics for a successful trial.

The GP control parameters set the maximum depth for rule-base trees in the initial population to six, the maximum depth of mutation subtrees to four, and the maximum rule-base depth after crossover to

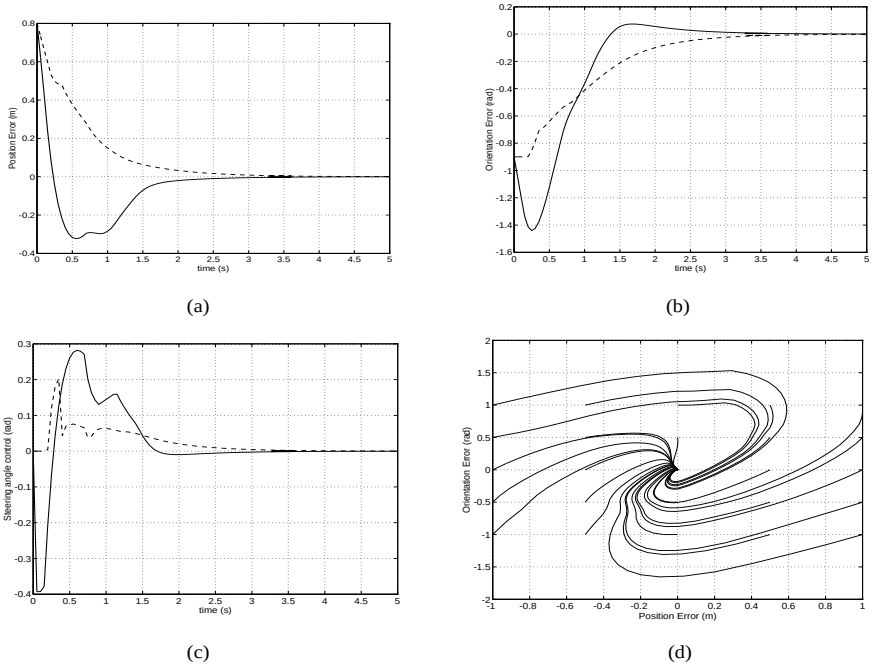

Figure 3: Performance comparison: GP-evolved FLC —, Hand-derived FLC - -; (a) ε_d, (b) ε_θ, (c) δ, (d) phase portrait.

seven. At each generation, breeding of the population was done using probabilities of 0.1 for reproduction, 0.5 for crossover (at any valid point), and 0.4 for mutation. The relatively high mutation rate (40%) was chosen as an attempt to compensate for any limitations that might stem from small population size. Tournament selection was used with a tournament size of two. In the run that yielded the best observed result (according to (5)), GP discovered the best rule-base after 7 generations; it had 21 rules, a raw fitness of 1.58 and 8 hits. Our hand-derived rule-base had 25 rules and scored a raw fitness of 1.96 and 8 hits. Based on a worst-case raw fitness of 72.5 for a given rule-base, these rule-bases correspond to 98% fitness and 97% fitness respectively.

For performance comparison, Figure 3(a–c) graphically illustrates the position error, orientation error, and control effort for the GP-evolved fuzzy controller and the hand-derived fuzzy controller. All results shown are for error category (d) of (2) with $\varepsilon_d = 0.8$ m and $\varepsilon_\theta = -0.9$ rad as initial conditions. Among the eight error categories, category (d) was shown [25] to be the most general for studying path tracking for tricycle-type vehicles. It is most general in the sense that in the process of correcting vehicle steering from initial error states in categories (a–c) (and corresponding symmetric cases), the vehicle error status ultimately reduces to category (d) or its counter-pair. We observe that the rise time of the GP-evolved controller is fastest and results in an overshoot (figure 3a) of the goal by about 0.3 m ($\approx$ 1 ft.) before it hones in on the path about 2 seconds later. Its response time in reaching the goal, however, is practically the same as for the hand-derived controller. The hand-derived controller forces the errors to zero in a smoother manner and without overshoot. However, it was explicitly designed to exhibit this type of behavior. Recall that the fitness measure used to drive the evolution of the GP-evolved rule-base favors rule-bases that result in small final errors. There was no selective pressure for rule-bases to exhibit smooth response without overshoot. Nonetheless, the results compare favorably with those of the hand-derived fuzzy controller. Observing the control efforts (figure 3c), we see that the steering angle for the GP-evolved controller spans a wider range of motion, and as a result, expends more energy in achieving

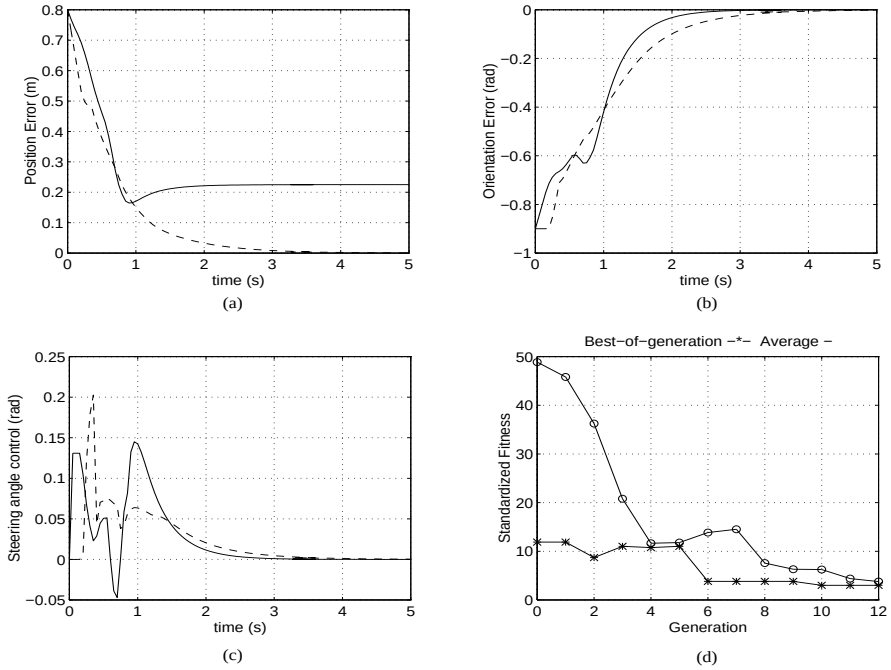

Figure 4: GP-evolved —, Hand-derived - -; (a) ε_d, (b) ε_θ, (c) δ, (d) standardized fitness curves.

the goal. Although the evolved controller learned the steering control rules using only 8 pre-selected fitness cases, it was able to generalize when started from initial conditions throughout the error state space. This is shown in the phase portrait of figure 3d which reveals that the origin is a stable node of the system. In other problems this may not be the case. In such situations it may be necessary to use random initial conditions in each fitness case to avoid overfitting pre-selected initial conditions. It should be noted that in this particular run the evolved rule-base resulted from the anomalous presence of a highly fit rule-base in the initial population whose genetic material persisted in the early generations and was only slightly improved by GP in generation 7. More dramatic evolutionary improvements over the generations were shown in other GP runs.

GP runs with different random seeds, population sizes, and numbers of generations yielded comparable performance results. In one instance, the GP discovered a rule-base of 30 rules that exhibited results very similar to those shown in Figure 3. Less control effort was expended and, consequently, the overshoot amplitudes were reduced. Other rule-bases of 21 rules were also found that can be considered to have performed as well as the hand-derived controller if a steady-state position error of 0.2 m ($\approx$ 8 inches) was acceptable according to the control system specifications. Results from one of these 21-rule controllers are shown in Figure 4. Generally, slightly faster settling times were observed with rule-bases of 21 rules. Figure 4d depicts a typical progression of the evolution process as a plot of standardized fitness vs. generations for the population average (–o–) and best-of-generation (–*–) rule-base. The response curves in Figures 4(a–c) are for the best-of-run rule-base which GP found in generation 10; it had a raw fitness of 2.8 and 7 hits. The result shown in the figure is for the only failed fitness case in which the steady-state position error exceeds the specified tolerance by about 0.05m ($\approx$ 2 inches).

A consolidated idea of the performance of GP for the steering control problem can be obtained from Figure 5. The figure shows the mean GP performance over six independent runs, each starting with distinct random number generator seeds. Mean performance is plotted as a fitness percentage (based on the worst-case raw fitness) for the best rule-base and population average rule-base in each generation.

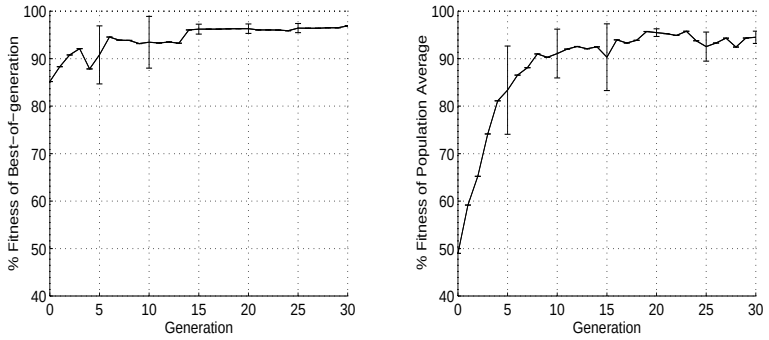

Figure 5: Mean performance of GP with state-error norm as fitness.

Error bars are shown every five generations indicating the performance variance one standard deviation from the mean. On average, GP was capable of improving the current best rule-base in the population (Figure 5a) throughout the evolution; the average rule-base in the population itself (Figure 5b) also improves. The best fuzzy rule-base in the initial population is 85% fit for path following in the plane. After 30 generations the best fuzzy rule-base is 97% fit. The presence of at least one 85% fit solution in the initial population suggests that finding a set of fuzzy rules for this problem is not very difficult. As such, methods like hill-climbing or even random search might do just as well. In any case, the problem reveals potential utility of GP for fuzzy rule-base evolution.

6.1 Results with modified fitness measure

The smallest rule-base discovered by GP had 11 rules and a raw fitness of 1.99 according to (5). However, the control signal was unacceptable due to fast oscillations of the steering angle during the first second or so of control. Such oscillations could cause damage to the robot's steering mechanism. This controller scored a fitness close to that of our hand-derived rule-base due to the "blindness" of the fitness measure to events taking place before the end of each fitness case. This revealed a necessity to modify the fitness measure in subsequent runs to include control effort as a cost. Several runs were executed after modifying the fitness measure to determine if it would induce the desired effect of minimizing the control effort. The following fitness function was used,

$$Raw fitness = \sum_{i=1}^{8} \sqrt{(\varepsilon_d^2 + \varepsilon_\theta^2 + \bar{\delta}^2)_i} \tag{6}$$

where $\bar{\delta}_i$ is the average control effort expended for fitness case i. Since lower raw fitness is associated with better rule-bases, this fitness function favors rule-bases that expend the least average control effort over the fitness cases. The best observed results after modifying the fitness measure are shown in Figure 6 along with the response curves for the complete rule-base. The GP evolved a new rule-base that had 18 rules, a raw fitness (according to (6)) of 1.37, and 8 hits. The fuzzy controller with the complete rule-base scored a raw fitness of 2.43 using (6). We see that there is indeed a reduction in the control effort (figure 6c) due to the selective pressure induced by (6). This reduction in control effort prevailed in *all* of the fitness cases. Thus, the new controller is the fittest despite the borderline, but acceptable, steady-state position error of 0.149m ($\approx$ 6 inches) for this single case. The importance of the fitness function used in evolutionary algorithms is evident in these examples. It is important that the fitness function map observable parameters of the problem into a spectrum of values that differentiate the

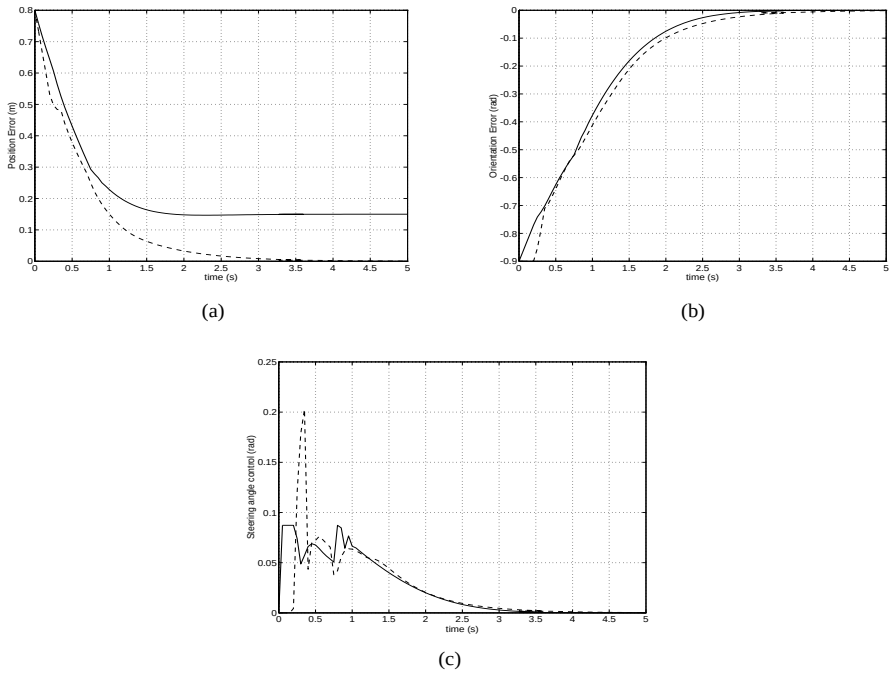

Figure 6: Performance comparison using control effort as a cost: GP-evolved —, Hand-derived - -; (a) ε_d, (b) ε_θ, (c) δ.

performance of individuals in the population. If the spectrum of fitness values is not sufficiently rich, the fitness function may not provide enough information to guide GP toward regions of the search space where improved solutions might be found. The most common types of fitness functions used in GP are error measures and problem-specific payoffs. For problems involving simulation of controlled behavior, a variety of performance attributes can be considered for inclusion in the fitness measure. Examples include a maximum number of time steps, explicit error tolerances, terminating physical events such as task success or failure, and penalties/rewards thereof. In general, selected performance attributes can be weighted to emphasize their importance in the search for candidate solutions. The fitness function is analogous to the performance measure of optimal control theory, or more generally, the objective function of optimization theory.

7 Conclusions

This investigation has revealed the potential of genetic programming as a tool for designing rule-bases for fuzzy logic controllers. For the purpose of evolving rule-bases, the GP implementation has some advantages over the simple GA and neural networks. Namely, it facilitates manipulation of the linguistic variables directly associated with the problem, and it allows for populations of rule-bases of various sizes. An additional feature of the syntactic structure is that it provides for context preservation as a by-product of structure-preserving crossover.

Good results have been obtained using small populations of rule-bases and the constrained syntactic structure for S-expressions. A number of fuzzy rule-bases have been evolved whose performances have been found to compare favorably with that of a complete rule-base derived by the authors. Several evolved rule-bases performed better than the human-derived solution according to respective fitness measures imposed on the simulated evolution. GP was able to produce fuzzy controllers that required

fewer rules than necessary for rule-base completeness. As with alternative evolutionary algorithms, the fitness function can be tailored to emphasize desiredperformance attributes. We found that the best runs were those that used tournament selection as opposed to fitness-proportionate selection.

Regions of the search space with favorable rule-bases were consistently found using GP. In many cases suboptimal solutions with respect to the objective fitness function were found, suggesting that GP performs well as a global adaptive search method. Possible improvements toward optimal solutions can be made by synthesizing a hybrid between GP and a localized search method such as hill-climbing [14]. From the vantage point of fuzzy rule-based systems design, initial results suggest that seeding initial rule-bases with prior knowledge (e.g. rules ensuring stability), and perhaps, additional tuning of fuzzy membership functions may be necessary to improve the robustness of the GP solutions. Additional modifications that may improve on the results reported here are: adding different inference and defuzzification methods as options to be selected by the evolutionary process, and using random initial conditions in each fitness case to avoid overfitting pre-selected initial conditions.

In future work we will investigate the evolution of larger populations for more generations. This future work will go a step further by challenging GP to scale up from evolving regulatory and tracking fuzzy behaviors to evolution of task-achieving fuzzy behaviors and rules for behavior coordination.

Acknowledgments:
The authors would like to thank Drs. Stephanie Forrest and Sharif Heger of the University of New Mexico for their encouragement, and for their roles in providing access to the computing resources used for this work. We also acknowledge commentary from the reviewers which was beneficial in making improvements to earlier drafts of the article.

References

[1] **Jamshidi, M., N. Vadiee and T. Ross** (Eds.) *Fuzzy Logic and Control: Software and hardware applications.* Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] **Koza, J.R.** *Genetic Programming: On the programming of computers by means of natural selection.* MIT Press, Cambridge, MA, 1992.

[3] **Mamdani, E.H. and S. Assilian** "An experiment in linguistic synthesis with a fuzzy logic controller", *International Journal of Man-Machine Studies* Vol. 7, pp. 1-13, 1975.

[4] **Karr, C.L.** "Design of an adaptive fuzzy logic controller using a genetic algorithm", *4th International Conference on Genetic Algorithms*, pp. 450-457, 1991.

[5] **Singleton, A.** "Genetic programming with C++", *BYTE Magazine*, pp. 171-176, February 1994.

[6] **Nordin, P.** "A compiling genetic programming system that directly manipulates the machine code", In Kinnear, K.E. Jr. (Ed.). *Advances in Genetic Programming*, MIT Press, Cambridge, MA., pp. 311-331, 1994.

[7] **Koza, J.R.** "Hierarchical automatic function definition in genetic programming", In Whitley, L.D. (Ed.). *Foundations of Genetic Algorithms* 2, Morgan Kaufmann, San Mateo, CA, pp. 297-318, 1993.

[8] **Zadeh, L.A.** "Fuzzy sets", *Information and Control* Vol. 12, pp. 338-353, 1965.

[9] **Homaifar, A. and E. McCormick** "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms", *IEEE Transactions on Fuzzy Systems*, Vol. 3, No. 2, pp. 129-139, May 1995.

[10] **Huang, S. and R.M. Nelson** "Artificial neural networks used as a rule selector for fuzzy logic controllers", *ASME Conference on Computers in Engineering*, pp. 445-454, 1993.

[11] **Feldman, D.S.** "Fuzzy network synthesis with genetic algorithms", *5th International Conference on Genetic Algorithms*, pp. 312-317, 1993.

[12] **Kinzel, J., F. Klawonn and R. Kruse** "Modifications of genetic algorithms for designing and optimizing fuzzy controllers", *1st IEEE Conference on Evolutionary Computation*, Orlando, FL, pp. 28-33, June 1994.

[13] **Goldberg, D.E.** *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[14] **O'Reilly, U.-M.** *An Analysis of Genetic Programming*, Ph.D. Thesis, School of Computer Science, Carleton University, Ottawa, Ontario, September 1995.

[15] **Cordón, O., F. Herrera, and M. Lozano** "A classified review on the combination fuzzy logic-genetic algorithms bibliography", Technical Report DECSAI 95129, Dept. of Computer Science and AI, University of Granada, Spain, October 1995.

[16] **Goldberg, D.E.** "Dynamic system control using rule learning and genetic algorithms", *9th International Joint Conference on Artificial Intelligence*, pp. 588-592, August 1985.

[17] **Grefenstette, J.J.** "A system for learning control strategy with genetic algorithms", *3rd International Conference on Genetic Algorithms*, pp. 183-190, 1989.

[18] **Schultz, A.C. and J.J. Grefenstette** "Improving tactical plans with genetic algorithms", *2nd International Conference on Tools for AI*, pp. 328-334, 1990.

[19] **Schultz, A.C.** "Learning robot behaviors using genetic algorithms", *1st World Automation Congress*, Vol. 1., pp. 607-612, August 1994.

[20] **Grefenstette, J.J.** "Lamarckian learning in multi-agent environments", *4th International Conference on Genetic Algorithms*, pp. 303-310, 1991.

[21] **Harvey, I., P. Husbands and D. Cliff** "Issues in evolutionary robotics", Technical Report CSRP 219, University of Sussex, England, July 1992.

[22] **Reynolds, C.W.** "Evolution of corridor following behavior in a noisy world", *From Animals to Animats 3: Third International Conference on Simulation of Adaptive Behavior*, MIT Press, pp. 402-410, August 1994.

[23] **Fraser, A.P. et al** "Evolving multiple agent behaviors for biologically inspired robots", In Brooks, R.A. and Maes, P. (Eds.). *Artificial Life IV*, MIT Press, July 1994.

[24] **Handley, S.** "The Genetic Planner: the automatic generation of plans for a mobile robot via genetic programming", *International Symposium on Intelligent Control*, pp. 190-195, 1993.

[25] **Hemami, A.** "Steering control problem formulation of low speed tricycle-model vehicles", *International Journal of Control*, Vol. 61, No. 4, pp. 783-790, 1995.

[26] **Tunstel, E.** "Coordination of distributed fuzzy behaviors in mobile robot control", *IEEE International Conference on Systems, Man and Cybernetics*, Vancouver, BC, pp. 4009-4014, October 1995.

[27] **D'haeseleer, P.** "Context preserving crossover in genetic programming", *1st IEEE Conference on Evolutionary Computation*, Orlando, FL, pp. 256-261, June 1994.