# Genetic Design of Fuzzy Controllers†

**Mark G. Cooper & Jacques J. Vidal**
University of California, Los Angeles
3531 Boelter Hall
Los Angeles, California 90024
*cooper@cs.ucla.edu* *vidal@cs.ucla.edu*

## Abstract

This paper considers the application of genetic algorithms as a general methodology for automatically generating fuzzy process controllers. In contrast to prior genetic-fuzzy systems which require that every input-output combination be enumerated, we propose a novel encoding scheme which maintains only those rules necessary to control the target system. The key to our method, demonstrated on the classic cart-pole problem, is to represent each fuzzy system as an unordered list of an arbitrary number of rules. Both the composition and size of the rule base evolve from an initial random state. By using a compact rule base representation, successful systems evolve quickly, increasing the likelihood of success when applied to more complex problems.

## 1. Introduction

The ability to translate information supplied in linguistic terms into a computer-usable form has made fuzzy logic systems popular for implementing complex process control. However, most only represent the "best guesses" of human experts, and can therefore be improved using feedback to *tune* the initial fuzzy rule bases. Automatic rule base generation has been attempted using gradient descent techniques,

† Presented at the Second International Conference on Fuzzy Theory and Technology; Durham, NC, October, 1993.

feedforward neural networks, radial basis function networks, fuzzy clustering, and genetic algorithms. These systems learn to control the targeted process through training—examples are presented and the system is modified based upon performance evaluation. Unfortunately, the number of rules must generally be specified *a priori*, and learning capacity is often small. We believe, however, that these limitations are not inherent to the problem, but rather lie in the details of the learning procedure employed.

To overcome these deficiencies, our approach uses a genetic algorithm to determine the number of rules in the fuzzy rule base as well as their composition. Additionally, a novel encoding scheme for the fuzzy rules results in a more compact rule base than used previously, allowing more complexity to evolve.

We begin by introducing the fuzzy inference model, followed by a brief description of genetic algorithms (for a complete treatment of genetic algorithms, see Goldberg [1989]), and of the physical system used for demonstration—the cart-pole problem. Following a review of two earlier genetic-fuzzy proposals, we describe our alternative approach, present experiment results, and discuss the properties and limitations of our technique.

### 1.1. The Fuzzy Associative Memory Model

Fuzzy rules typically appear in the form "If $x_1$ is $v_1$ and $x_2$ is $v_2$ and..., then $a_1$ is $w_1$ and $a_2$ is $w_2$ and..." where $x_n$ are input variables, $a_n$ are output variables, and $v_n$ and $w_n$ are descriptors like "small positive", "large negative", "medium", etc. Associated with each descriptor is a *membership function* specifying the degree to which an input satisfies the descriptor. A *Fuzzy Associative Memory* (or FAM) (Kosko, 1992) applies these rules to a set of inputs, combines the consequents of each rule, and produces a value for each output variable. To illustrate this process, consider the following rule base (excerpted from Kosko):

1. If the pendulum angle, θ, has a small positive value (PS) and the angular velocity, Δθ, is near zero (ZE), then adjust the motor current, ν, by a small negative value (NS).

2. If the pendulum angle, θ, is near zero (ZE) and the angular velocity, Δθ, is near zero (ZE), then adjust the motor current, ν, by a value near zero (ZE).

These rules are displayed graphically in Figure 1. The top three graphs correspond to the first rule, and the bottom three correspond to the second. The first two columns (from left to right) represent the antecedent clauses, while the third represents the consequent clause.

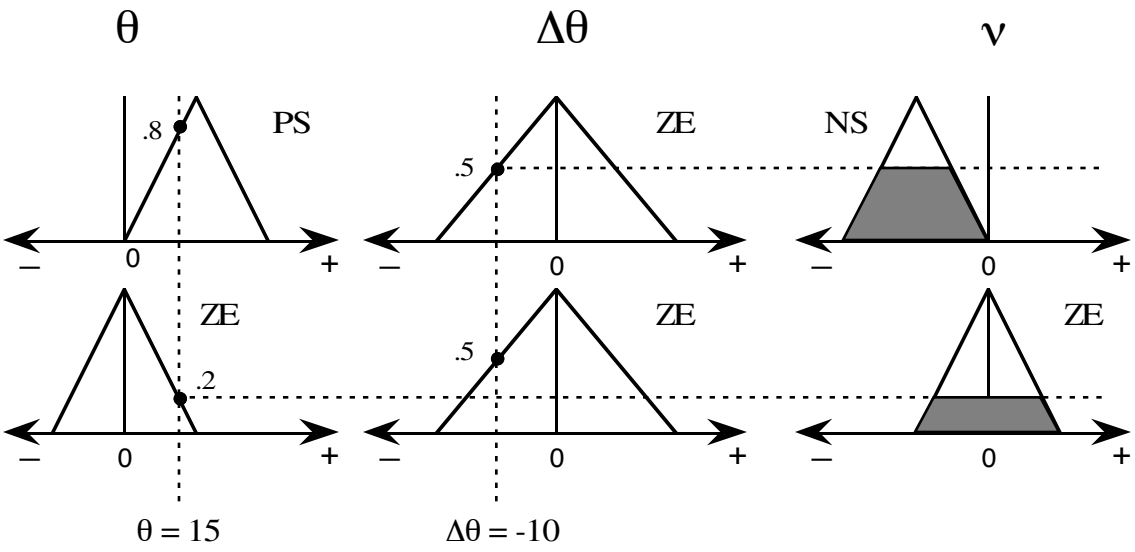


**Figure 1.** Fuzzy Reasoning (from Kosko)

Instantiations of the input variables are presented to the rule base in parallel. The membership function for each antecedent clause is applied to its corresponding input to produce a fit value for the clause. When the rule antecedent is made up of a conjunction of clauses, the minimum fit value among the clauses is taken as the fit for the entire rule, i.e., as the degree of applicability of the rule to the overall inference. In the example, the input values θ = 15 and Δθ = -10 are applied to the rule base with the antecedent fit values of 0.8 and 0.5 for the first rule,

and 0.2 and 0.5 for the second. The minimum of each pair yields the overall fit for each rule.

Next, the consequent of each rule is calculated as the *region* under its membership function below the antecedent fit value. These regions constitute the fuzzy output of the FAM. The output of the first rule is the region of the consequent's membership function below the antecedent fit value 0.5. Similarly, the output of the second rule is the region under 0.2.

The collection of regions generated as the output of the fuzzy rules is the qualified output of the FAM. To generate a single value for each consequent variable, these regions must be combined. This process, called *defuzzification*, is generally accomplished by overlaying the output regions and computing the centroid (Figure 2).

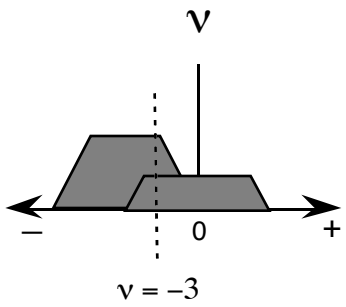


**Figure 2.** Defuzzification by Computing the Centroid

### 1.2. Genetic Algorithms

Genetic algorithms are probabilistic search techniques that emulate the mechanics of evolution. They are capable of globally exploring a solution space, pursuing potentially fruitful paths while also examining random points to reduce the likelihood of settling for a local optimum. The system being evolved is encoded into a long bit string called a *chromosome*. Sites on the chromosome correspond to specific characteristics of the encoded system and are called *genes*. Initially, a random set, or *population*, of these strings is generated. Each string is then evaluated according to a given performance criterion and assigned a fitness score. The strings with the best scores are used in the reproduction phase to produce the next generation.

The reproduction of a pair of strings proceeds by copying bits from one string until a randomly triggered *crossover* point, after which bits are

copied from the other string. As each bit is copied, there is also the probability that a *mutation* will occur. Mutations include inversion of the copied bit, and the addition or deletion of an entire rule (Figure 3). These latter two mutations permit the size of a system's rule base to evolve. The cycle of evaluation and reproduction continues for a predetermined number of generations, or until an acceptable performance level is achieved.

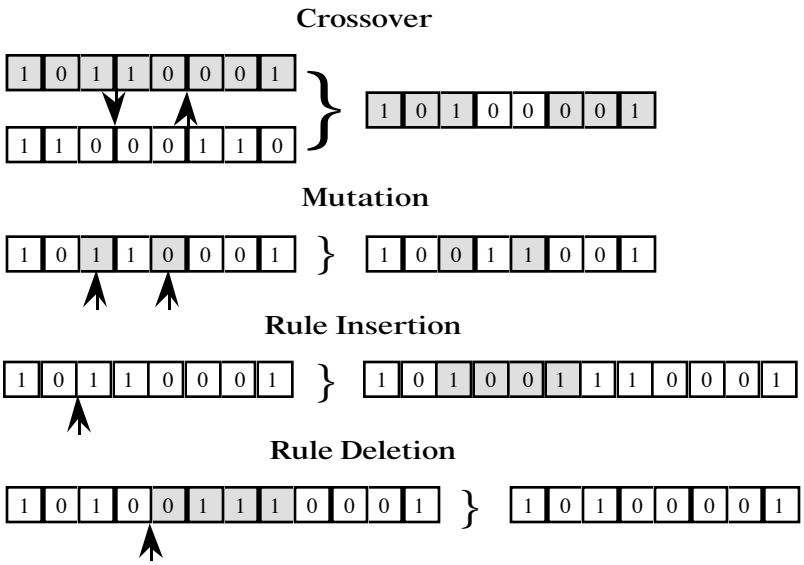


**Figure 3.** Genetic Operators

### 1.3. Pole Balancing

We will apply the evolved fuzzy systems to a classic control problem variously referred to as the "cart-pole", "inverted pendulum", or "pole balancing" problem. This problem is an example of an inherently unstable dynamic system, and is an established literature benchmark for evaluating control schemes. The objective is to control translational forces that position a cart at the center of a finite width track while simultaneously balancing a pole hinged on the cart's top (Figure 4). The physical plant can be simulated from a set of nonlinear differential equations describing the cart and pole dynamics (Wieland, 1991).

$$\ddot{x} = \frac{F - \mu_c \, sign(\dot{x}) + \tilde{F}}{M + \tilde{m}} \tag{1}$$

$$\ddot{\theta} = -\frac{3}{4l}\left(\ddot{x}\cos\theta + g\sin\theta + \frac{\mu_p \dot{\theta}}{ml}\right) \tag{2}$$

where

$$\tilde{F} = ml\dot{\theta}^2 \sin\theta + \frac{3}{4} m \cos\theta\left(\frac{\mu_p \dot{\theta}}{ml} + g\sin\theta\right) \tag{1a}$$

$$\tilde{m} = m\left(1 - \frac{3}{4}\cos^2\theta\right) \tag{1b}$$

**Table 1.** Symbols Used in Equations 1 and 2

| Symbol | Description | Value |
|---|---|---|
| $x$ | Cart position | [-1.0, 1.0] m |
| $\theta$ | Pole angle from vertical | [-0.26, 0.26] rad |
| $F$ | Force applied to cart | [-10, 10] N |
| $g$ | Force of gravity | 9.8 m/s$^2$ |
| $l$ | Half-length of pole | 0.5 m |
| $M$ | Mass of the cart | 1.0 kg |
| $m$ | Mass of the pole | 0.1 kg |
| $\mu_c$ | Friction of cart on track | 0.0005N |
| $\mu_p$ | Friction of pole's hinge | 0.000002 kg m$^2$ |

The cart and pole are initially placed at rest at a predetermined position. A simulation succeeds when sixty seconds of simulated time elapse without the cart reaching the end of the track or the pole falling over. For our purposes, we consider a pole angle of 0.26 radians (about 14 degrees) the point at which the pole "falls over." Wieland (1991) investigates the effect of varying the angle at which a failure occurs on the process of evolving a neural network that controls the cart-pole system and concludes that increasing this parameter does not necessarily generate superior solutions, and makes training slower.

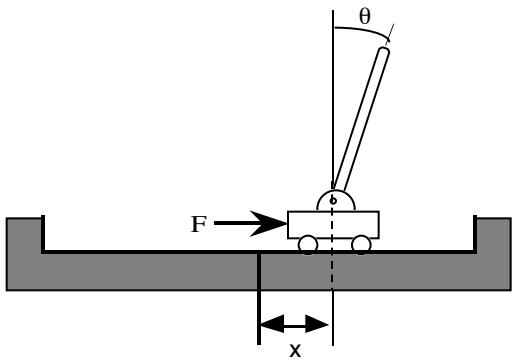


**Figure 4.** The Cart-Pole System

## 2. Generating Fuzzy Systems Using Genetic Algorithms

A central issue in fuzzy control is the selection and encoding of the fuzzy rules. Sufficient system information must be encoded and the representation must be amenable to evaluation and, in the genetic framework, reproduction. The representation must allow flexibility both in the number and content of the discovered rules. Finally, a proper scoring function is essential. We will describe two earlier applications of genetic algorithms to the automatic generation of a fuzzy rule base, and compare these to our approach.

One of the earliest applications of genetic algorithms to the design of fuzzy systems was developed by Karr (1991), again to solve the cart-pole problem. The production of the fuzzy controller begins with the definition of the fuzzy sets used to describe each input variable. Each of the four input variables are characterized by three fuzzy sets—NEGATIVE, ZERO, and POSITIVE—yielding 81 possible combinations. The fuzzy system designer then assigns one of seven choices for the output to each input combination. The resulting fuzzy system represents the expert's "best guess". The membership function extrema are then encoded into a bit string, and a genetic algorithm is applied to shift the membership functions so as to find locations which improve performance. The evolved system consistently outperforms the original, being capable of recovering from initial positions that fail under the original rule base.

Lee & Takagi (1992) improve on Karr's method by permitting both the size and the composition of the rule base to emerge as the result of the genetic search, instead of being specified by the system designer.

However, the maximum number of fuzzy sets permitted for each variable is still fixed. Each membership function specification consists of center, right base, and left base values, each requiring one byte of storage. Each output variable has a three-byte description of the action to be taken in response to every input variable combination. There are $\prod m_i$ descriptions associated with each output variable, where $m_i$ is the maximum number of membership functions permitted for input variable $i$. Therefore, the byte length of the entire string is $3\ (\sum m_i + v\ \prod m_i)$, where $v$ is the number of output variables. In their example, each input variable is assigned a maximum of 10 membership functions, and each system description requires 360 bytes. Hence, 30,120 bytes are required to encode the four-input version of the problem. As the number of variables increases the length of the string encoding the system size increases exponentially, with a corresponding exponential increase in the complexity of the search space. Such a system is unlikely to scale well for more complex problems.

## 3. A Compact Rule Base Approach

### 3.1. System Representation

It is a long recognized limitation of genetic algorithms that, as a global search technique, the encoded bit string lengths should be kept as small as possible as they evolve, since the size of the search space increases exponentially with the size of the strings. Therefore, it is highly desirable to maintain only the rules necessary to accomplish the task. This is the goal pursued in our approach. The two evolutionary requirements listed in Lee & Takagi are also maintained; namely, that the number of rules needed to accomplish the task must be evolved, and that the membership functions comprising those rules must evolve from a random initial state.

Our version of the cart-pole problem has four input variables (x-position, θ-position, dx/dt, dθ/dt), and one output variable (the force applied to the cart). The membership function for each variable is a triangle characterized by the location of its center and the half-length of its base. A single rule, therefore, consists of the concatenation of ten one-byte unsigned characters (assuming values from 0 to 255) specifying the centers and half-lengths of the membership functions (Figure 5). The rule

descriptions for a single fuzzy system are then concatenated into a single bit string where the number of rules is not restricted.

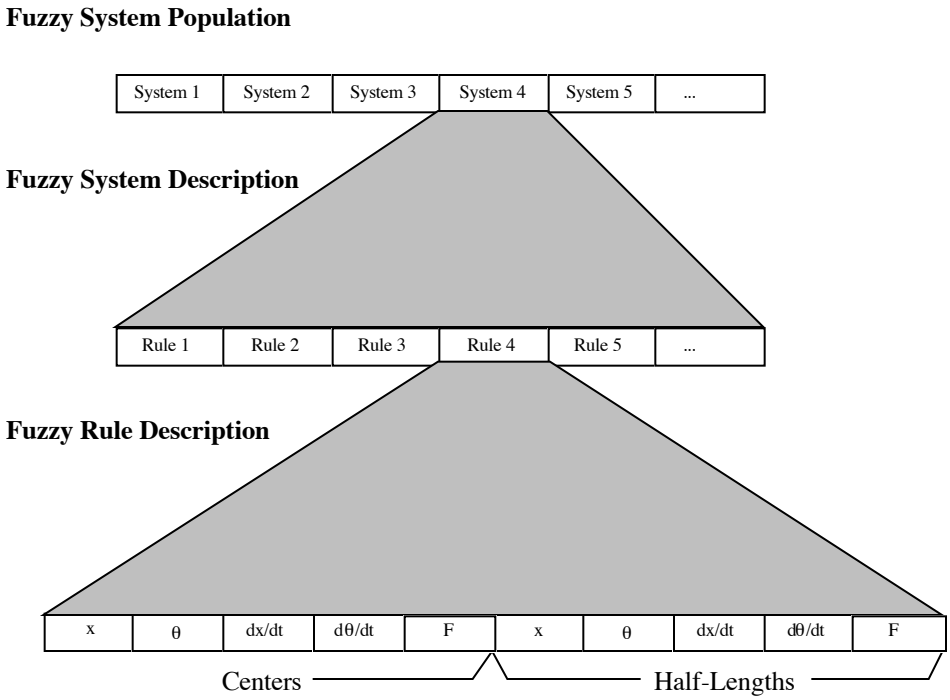


**Figure 5.** Organization of the Fuzzy System Population

One limitation of earlier attempts at fuzzy system learning was the assumption that each rule is dependent upon combinations of all of the input variables. In many cases this is undesirable since the appropriate action is often strongly conditioned by only a small subset of the variables. Without the ability to ignore irrelevant variables, rules evolve which account for every combination of spurious variables with each required value for the relevant ones. We address this problem by ignoring a variable when its corresponding half-length falls outside of a particular range ([40, 215] out of the overall range of [0, 255] in our experiments).

### 3.2. System Evaluation

Each evolved fuzzy system is evaluated according to the length of time that it keeps the cart-pole system from failing. Twenty different initial starting positions are considered, arranged symmetrically around

{x = 0, θ = 0} (i.e., if the position {x = 0.5, θ = 0.07} is in the test suite, so too are {0.5, -0.07}, {-0.5, -0.07}, and {-0.5, 0.07}). The selection of a test suite representative of all possible combinations of input values is critical for evolving a robust system.

Each fuzzy controller's objective is to keep the pole balanced and the cart positioned near the center of the track. The score for a particular trial accumulates as long as the system does not fail. Once a failure occurs the next test immediately begins. Performance scores, however, need only be based upon the deviation of cart's position from the center. The pole's angle is not required explicitly in the scoring equation since it is already an implicit constraint on the score—once the pole angle exceeds a critical value the simulation is terminated. In fact, when both the x-position angle are included in the scoring equation, the location of the cart is almost entirely ignored whereas including only the x-position in the scoring equation, both variables assume comparable importance.

$$score = \sum_{20\,\text{test cases}} \sum_{60\,\text{seconds}} \sum_{100\,\frac{\text{updates}}{\text{second}}} \left( 1.0 - \frac{x}{x_{\max}} \right) \tag{3}$$

### 3.3. Rule Alignment and Reproduction

To be meaningful, the genetic paradigm requires that the rules in the two strings be aligned so that similar rules are combined with each other. Consider, for example, two fuzzy systems having the same rule set, but arranged differently in each. Simply combining the strings in the order they appear does not preserve much information about either system and produces nearly random results, rather than a child system that performs in a manner similar to its parents. Therefore, before reproduction, the two strings must be aligned so that the centers of the input variables match as closely as possible. In our algorithm, the most closely matching rules (defined as the sum of the differences between the input variable centers) are combined first, followed by the next most closely matching rules from those that remain, and so on (Figure 6). Any rules from a longer string that are not matched are added at the end. This matching scheme is not optimal, but requires considerably less time and performs nearly as well as the optimal method which involves generating

the difference matrix between each pair of rules, then solving the "distribution problem" to find the minimal matching.

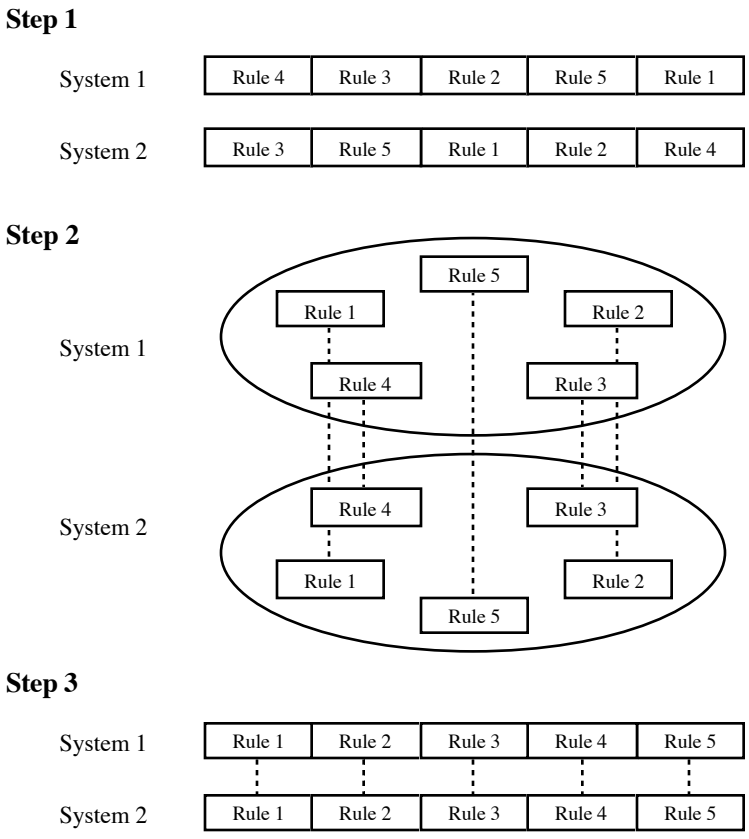


**Figure 6.** Rule Alignment

## 4. Simulation Results

To demonstrate our training procedure, several fuzzy cart-pole controllers were evolved. The population size in each trial was 200, with each system randomly assigned between one and thirty rules initialized with random characters. The fitness of each controller was then established according to its performance on the cart-pole system for sixty simulated seconds, with the state of the system updated using Euler's method 100 times every second. Pairs of strings from the fifty top scoring controllers were randomly selected and combined to produce the next generation (the genetic parameters appear in Table 2).

**Table 2.** Genetic Parameters Used in Fuzzy Rule Base Generation

| | |
|---|---|
| Crossover Rate: | 0.0113 |
| Mutation Rate: | 0.0115 |
| Add Rule Mutation Rate: | 0.0040 |
| Delete Rule Mutation Rate: | 0.0040 |

Learning typically occurred quickly. By about the fiftieth generation the top-scoring controller was usually capable of maintaining the cart-pole system for the entire sixty seconds (as opposed to the 5000 generations required in Lee & Takagi). The progress of training for several runs appears in Figure 7.

To illustrate the results of a typical run in detail, an evolved rule base appears in Figure 8. Each row corresponds to a single rule, and each column corresponds to a system variable—x, $\theta$, dx/dt, d$\theta$/dt, and F respectively. This rule base was then applied to a random initial position not used during training ($x = 0.54$, $\theta = -0.077$). Traces of the x-position, $\theta$-position, and force applied appear in Figure 9. The controller does not actually bring the system to rest, but rather establishes a stable oscillation. In fact, the fuzzy controller maintains the cart-pole system upright through 24 hours of simulated time. It is impossible to bring the cart-pole system completely to rest (the oscillations just keep getting smaller and smaller). In our example, the closest that the system comes to being centered at rest after 24 hours was $x = 0.000067$, $\theta = 0.00085$, $dx/dt = 0.000011$, and $d\theta/dt = 0.000027$.

## 5. Discussion

The most sigificant contribution of our approach is the ability to evolve a fuzzy rule base consisting solely of relevent rules, in contrast to previous approaches which require that every possible combination of a fixed set of inputs be ennumerated, or require that the relevancy of a large suite of rules be determined. Since large systems are generally more difficult to evolve, a selection pressure inherent to the genetic algorithm method, no explicit selection pressure is required to reduce the number of rules in the evolved controllers (as Lee & Takagi [1993]).

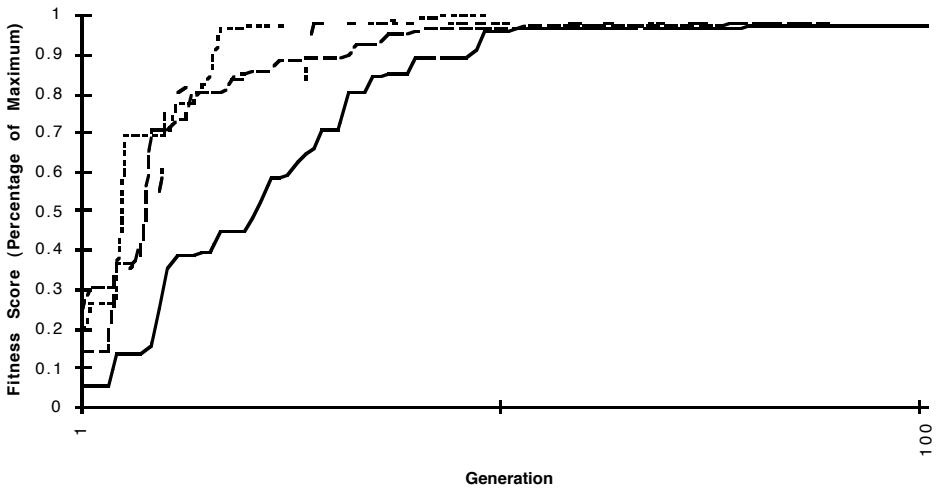


**Figure 7.** Progress of Training for Several Evolutionary Trials

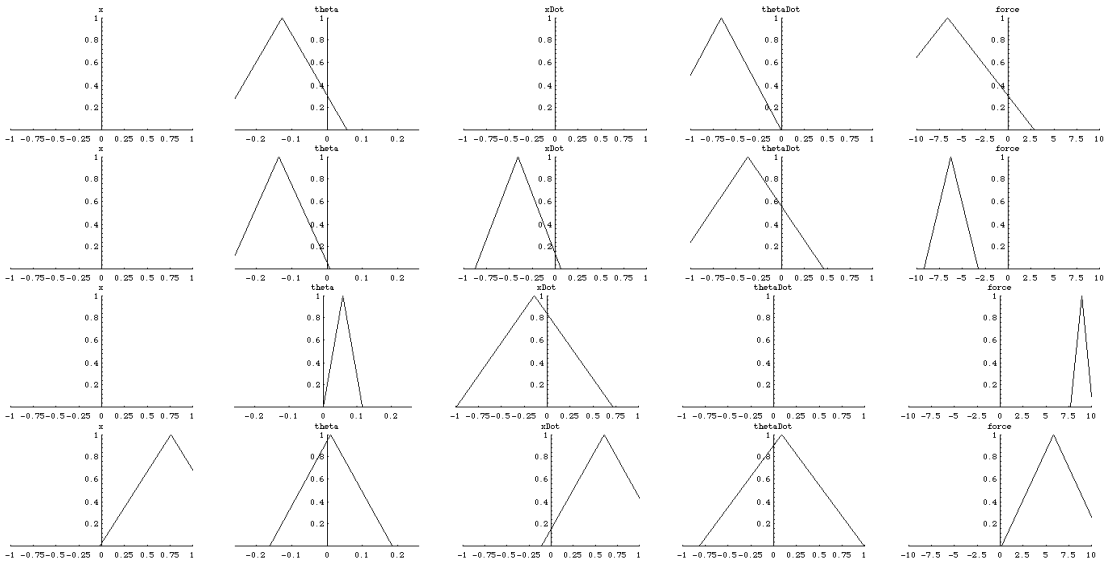


**Figure 8.** Fuzzy Rule Base Produced by the Genetic Algorithm

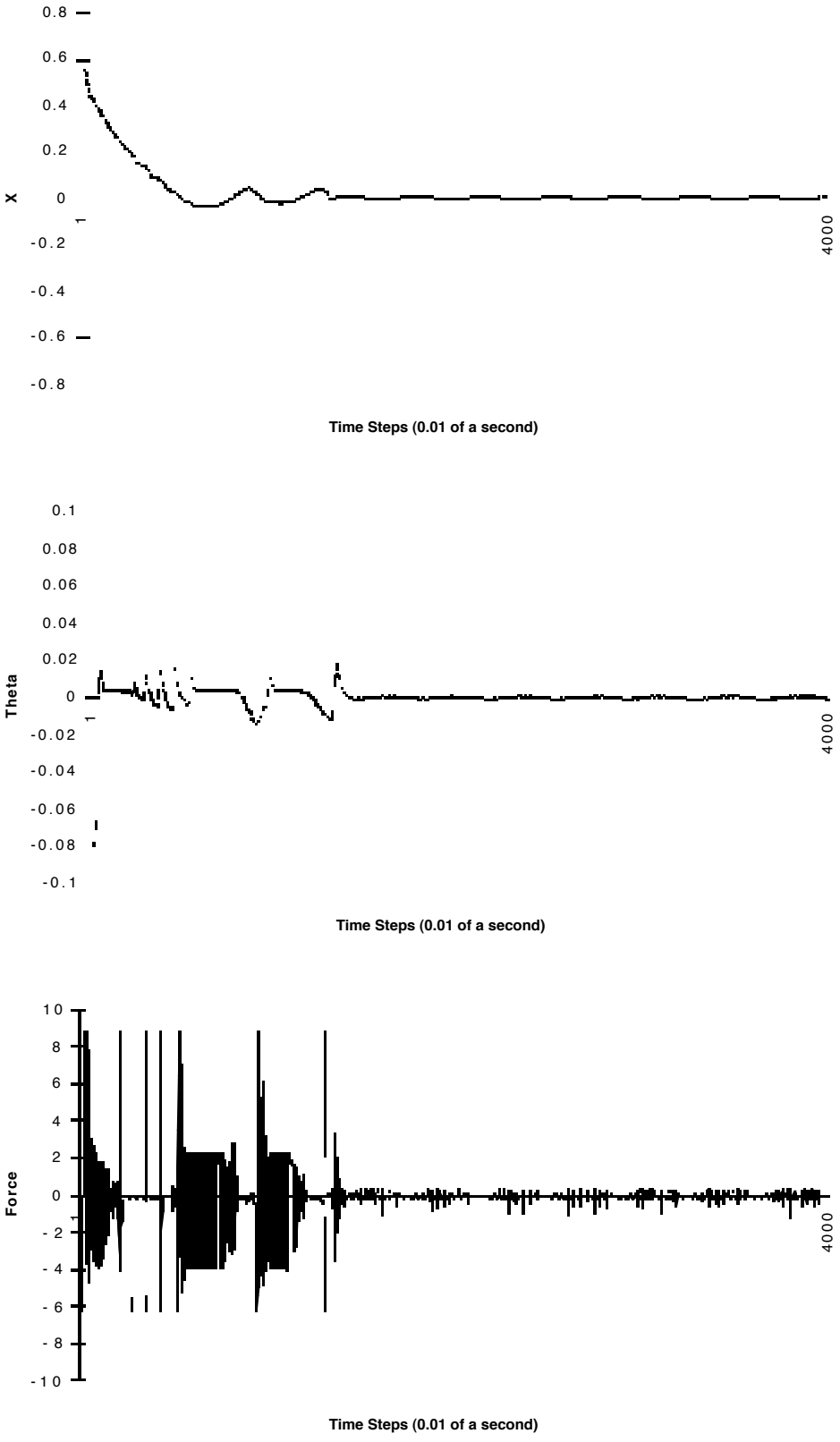


**Figure 9.** Traces of Selected Variables in the Trial Run

The fuzzy controllers evolved by our method, like most systems generated using genetic algorithms, are very sensitive to the selection pressures imposed by the system designer. These include the training set and the performance criteria. With regard to the composition of the training set, the range and distribution of the test cases must be representative of the expected system input. If the test cases are unevenly distributed, e.g., most of the initial positions have positive x-positions while very few have negative x-positions, then the evolved systems will fail to learn the correct control action for the rarer cases. In fact, continued training after learning to correctly handle the common cases still fails to improve performance. This is because in early generations the systems performing correctly on the common cases are more likely to reproduce because they perform well on a higher percentage of the training set. In later generations, the evolved systems cannot change sufficiently to successfully handle the rarer cases without degrading performance to the point where they are not selected to reproduce. Hence, the rarer cases are never adequately learned.

Additionally, careful consideration must be made in choosing a performance criterion. In trials where the divergence of both x and $\theta$ were considered in the scoring equation, rule bases were generated which successfully balanced the pole, but which ignored the x-position of the cart. Since success in balancing the pole is the primary prerequisite for the continuance of each test case, the x-position became an unimportant evolutionary factor and often ignored. Therefore, we used the x-position as an explicit training criterion while the $\theta$-position remained an implicit training criterion.

Despite the fact that the rule bases generated by this method include only those rules required to solve the problem, there are instances where rules are duplicated, or where one rule is subsumed by another. One direction for continued research is to be able to automatically identify instances where rules can be combined or eliminated. This will allow the rule bases to be further consolidated. Due to the compactness of the representations, we expect that this method will be capable of addressing more complex problems than previous approaches. Our future research will more fully characterize this method by observing the effects of

changing system parameters (e.g. pole length) during evolution and by examining in detail the problem solving methodologies discovered by the genetic search. We will also attempt more difficult problems such as the multiple pole and the jointed pole systems to expand the limits of genetic-fuzzy system generation.

In short, the fuzzy systems produced by genetic search are extremely faithful to both the training set and the performance criteria used during evolution. In many cases, the resulting systems perform in a manner consistent with these parameters, yet still differ from the vision of the system designer. Therefore, although the objective of automatic fuzzy system generation is to eliminate the need for a human expert, learning solely through the observation and feedback, an expert is still required to shape the progress of learning, and to interpret and evaluate the resulting systems.

## References


Karr, C.L. (1991) "Design of a Cart-Pole Balancing Fuzzy Logic Controller Using a Genetic Algorithm", *Proceedings of the SPIE Conference on the Applications of Artificial Intelligence* . Bellingham, WA: SPIE—The International Society for Optical Engineering, pp 26-36.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning* . Reading, MA: Addison-Wesley.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

Lee, M.A. & H. Takagi (1993) "Integrating Design Stages of Fuzzy Systems using Genetic Algorithms", *Proceedings of the Second IEEE International Conference on Fuzzy Systems.* New York, NY: Institute of Electrical and Electronics Engineers, pp 612-617.

Wieland, A.P. (1991) "Evolving Controls for Unstable Systems", In S. Touretzky et. al. (ed.), *Connectionist Models: Proceedings of the 1990 Summer School.* San Mateo, CA: Morgan Kaufmann Publishers, Inc., pp 91-102.