Genetic Design of Fuzzy Controllers: The Cart and Jointed-Pole Problem

Mark G. Cooper & Jacques J. Vidal
University of California, Los Angeles
4531 Boelter Hall
Los Angeles, California 90024
cooper@cs.ucla.edu *vidal@cs.ucla.edu*

This paper considers the application of genetic algorithms to the automatic generation of fuzzy process controllers. In contrast to prior genetic-fuzzy systems which require that every input combination be enumerated, we employ a novel encoding scheme which maintains only those rules necessary to control the target system. The key to this method is to represent each fuzzy system as an unordered list of an arbitrary number of rules. This compact rule base *methodology, introduced in [1] to generate a controller for the cart and single-pole problem, is applied to the more complex cart and jointed-pole problem. By using a compact rule base representation, successful systems evolve quickly, increasing the likelihood of success when applied to complex problems.*†

1. Introduction

Fuzzy logic is becoming a increasingly popular component in the implementation of process control. Control information supplied by a human expert in linguistic terms is translated into a computer-usable form. In many respects, fuzzy logic provides a framework for formalizing intuition and common sense. However, process controllers and expert systems generated in this manner represent only the "best guesses" of human experts. An alternative approach is to generate the fuzzy system automatically, without relying on the human expert, by using feedback to repeatedly tune an existing rule base. Automatic rule base generation has been attempted using gradient descent techniques, feedforward neural networks, radial basis function networks, fuzzy clustering, and genetic algorithms. In all cases, examples are presented and each system is modified based upon an evaluation of its performance. Unfortunately, nearly all automatic rule base generation algorithms require that the number of rules be specified *a priori* and often have a very limited learning capacity. We believe, however, that these limitations are not inherent to the problem, but rather lie in the details of the learning procedure employed.

Although several researchers have investigated genetic-fuzzy systems specifically, our algorithm improves upon existing approaches in two ways. First, both the number of rules in the rule base and their composition emerge as the result of a genetic search without placing any arbitrary restrictions on the number of rules. Second, by using a compact rule base representation, the range of problems addressable by this method is much greater than that of many existing methods.

We begin by introducing the fuzzy inference model, genetic algorithm, and the physical system used for this demonstration. We then describe in detail our approach to evolving fuzzy systems, present experimental results, and finally discuss the properties and limitations of our technique.

1.1. The Fuzzy Associative Memory Model

Fuzzy rules typically appear in the form:

$$\texttt{If } x_1 \texttt{ is } v_1 \texttt{ and } x_2 \texttt{ is } v_2 \texttt{ and...,}$$
$$\texttt{Then } a_1 \texttt{ is } w_1 \texttt{ and } a_2 \texttt{ is } w_2 \texttt{ and....}$$

where x_n are input variables, a_n are output variables, and v_n and w_n are descriptors like "large positive", "small negative", "near zero", etc. Associated with

† Submitted for Presentation at the Third IEEE International Conference on Fuzzy Systems; Orlando, Florida, June 1994.

each descriptor is a *membership function* which specifies the degree to which a given input satisfies the descriptor. Sets of fuzzy rules are evaluated and combined according to the *Fuzzy Associative Memory (*or *FAM)* model described in [5].

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.

Next, the consequent of each rule is calculated as the region under its membership function below the antecedent fit value. The collection of regions generated as the output of the fuzzy rules constitutes the qualified output of the FAM. To produce 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.

1.2. Genetic Algorithms

Genetic algorithms are probabilistic search techniques that emulate the mechanics of evolution [2]. They are capable of globally exploring a solution space, pursuing potentially fruitful paths while also examining additional 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*. Each feature of the system located at a specific position in the string is called a *gene*. Initially a large random set, or *population*, of strings is generated. The genetic search begins by evaluating each string with respect to some task and assigning a fitness score. Then 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. As each bit is copied, there is also the probability that a modification or *mutation* will occur. Mutations include the inversion of a bit, or the insertion or deletion of an entire rule. Figure 1 illustrates these operations.

1.3. Jointed-Pole Balancing Problem

The task to which we will apply the evolved fuzzy systems is an extension of the classical non-linear control problem variously referred to as the "cart-pole", "inverted pendulum", or "pole balancing" problem. The objective is to control the translational

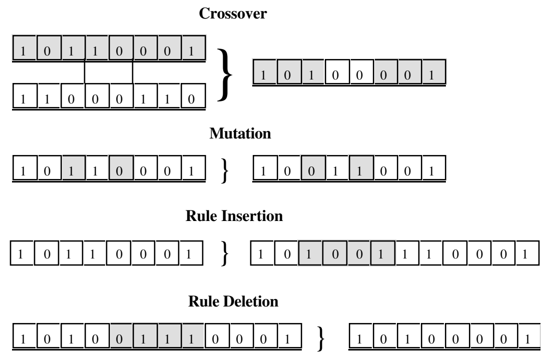

Figure 1. Genetic Operators

forces that position a cart at the center of a finite width track while simultaneously balancing a jointed pole hinged on the cart's top (Figure 2). This system is harder to control than the more common single-pole system (which uses five control variables) since the controller must consider the motion of two poles, and requires two additional control variables. The physical plant is simulated from a set of nonlinear differential equations describing the cart and pole dynamics [7].

The cart and poles are initially placed at rest in a predetermined configuration. A simulation succeeds when thirty seconds of simulated time elapse without the cart reaching the end of the track or one of the poles 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 [7] investigates the effect of varying the angle at which a failure occurs while evolving neural network cart-pole controllers and concludes that increasing this parameter does not necessarily generate superior solutions, and makes training slower.

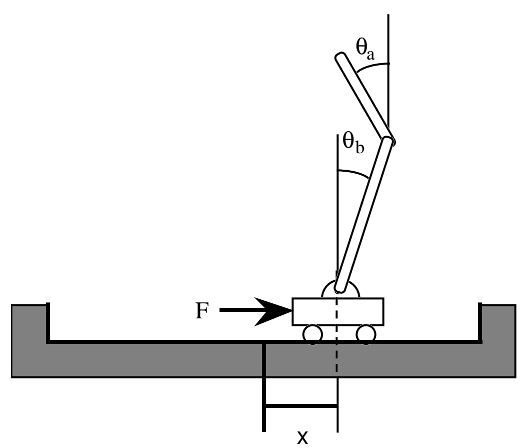

Figure 2. The Jointed Cart-Pole System

2. Methods

2.1. Prior Methods

In one of the first applications of genetic algorithms to fuzzy controller design [1], production of a controller for the cart and single-pole problem begins with the definition of several fuzzy sets to linguistically describe the input and output variables. The fuzzy system designer then, upon the advice of an expert, assigns an element from the output set to each input combination. The membership function extrema are then adjusted using a genetic algorithm to find locations which improve performance. However, the size of the rule base remains fixed and its composition must be initially defined by a human expert.

This method is improved upon in [6] by permitting the size of the rule base to emerge as the result of a genetic search (up to a preset maximum granularity for each rule), and the composition of the rule base to emerge from a random initial state. The binary representation for each combination of the input variables are concatenated into a single string which is then trained. The length of the bit string is therefore exponential in the maximum number of membership functions associated with the variables and hence would not be expected to scale well. This exponential increase in string size with problem complexity is also evident in [3].

2.2. System Representation

The selection and encoding of the rule base is the central issue in evolving fuzzy controllers. Sufficient information must be encoded in the bit string to reconstruct the fuzzy system, and the representation must be amenable to evaluation and, in the genetic framework, reproduction. The representation must also allow flexibility both in the number and content of the discovered rules. It has long been recognized that because genetic algorithms perform a global search of a solution space, the encoded bit string lengths should be kept as short as possible, since the size of the search space increases exponentially with string size. Many prior approaches require complete coverage of the input domain and consequently include rules which may be superfluous or unnecessary. Additionally, these methods make training more difficult as the search space becomes increasingly large.

In our *compact encoding scheme*, each of the seven control variables (x, θ_a, θ_b, dx/dt, $d\theta_a$/dt, $d\theta_b$/dt, and F) require two one byte integers (assuming values from 0 to 255)—one placing the center and the other specifying the half-length of the base of the isosceles triangle representing the membership function. In

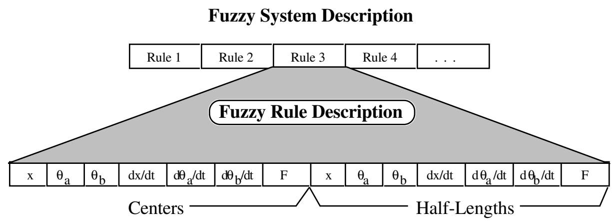

Figure 3. Fuzzy Rule Base Representation

general, each rule requires 2v bytes where v is the total number of control variables used by the system. The rule base for a fuzzy system is the concatenation of an unlimited number of individual rule strings (Figure 3).

Another issue that we address in our encoding scheme is that the action prescribed by each rule may not be dependent upon combinations of all of the input variables. In many cases the appropriate action may be strongly conditioned by only a small subset of the variables. Without the ability to ignore irrelevant variables, rules must evolve which account for every combination of spurious variables, each having the same values for the relevant ones. We have therefore included a mechanism by which variables are ignored when their half-length values fall outside of a certain range.

2.3. System Evaluation

Each evolved fuzzy system is evaluated according to the length of time that it prevents the cart-pole system from failing and on the distance of each input variable from zero. In other words, the closer the physical system is to being at rest, the higher the fuzzy system scores. Ninety-six initial starting positions are used to evaluate each fuzzy system. The selection of the test set is critical to the successful evolution of the controllers, and must be representative of all possible input combinations[1].

The score for a fuzzy system accumulates, as long as the simulation does not fail, according to the following equation:

$$\text{score} = \sum_{\substack{96\,\text{test} \\ \text{cases}}} \sum_{\substack{3000 \\ \text{updates}}} \sum_{\substack{6\ \text{input} \\ \text{variables}}} \left(1 - \frac{x_v}{x_v^{max}}\right)$$

where x_v is the value of an input variable and x_v^{max} is the maximum value for that variable. When a failure occurs, the test is immediately terminated and the next test begins.

2.4. System Reproduction

One of the most significant obstacles to a methodology for generating fuzzy controllers using a *rule-based* approach (as opposed to the prior *domain-based* approaches) is in reproduction. To be meaningful, the genetic paradigm requires that the two reproducing strings be aligned so that similar genes are combined with each other. In nature it would not make much sense for the mother's gene for good vision to combine with the father's gene for curly hair. Similarly here, directly combining two rule bases containing the same rules arranged in different orders, would certainly produce an undesirable result. This pitfall is not encountered in other approaches since all input combinations are exhaustively enumerated and the semantics of a particular position in the bit string can be calculated. Therefore, before reproduction, the rules in the two strings must be aligned so that they 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 4). Any rules that are not matched are appended to the end of the string. While this matching scheme is not optimal, it requires considerably less time and in most cases performs nearly as well as the optimal solution.

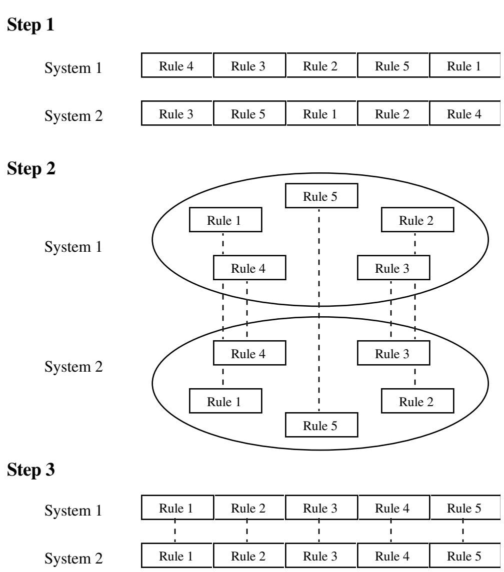

Figure 4. Fuzzy Rule Alignment

3. Results

To demonstrate our training procedure, several controllers were evolved. The population size in each trial was 200, with each system randomly assigned between one and thirty rules initialized with random strings. The fitness of each controller was established based on its performance on the physical system for thirty simulated seconds, with the state of the system updated using Euler's method 100 times every second. Pairs of strings from the top fifty scoring controllers were randomly selected and combined to produce the next generation (the genetic parameters appear in Table 1). Additionally, the best-scoring system was copied intact into the next generation to ensure that the high scores would be strictly non-decreasing throughout training.

Table 1. Genetic Parameters Used in Fuzzy Rule Base Generation

Crossover	0.005
Mutation	0.010
Add Rule	0.100
Delete Rule	0.100

Learning typically occurred quickly. By about the thirtieth generation, the top-scoring controller was capable of maintaining the cart-pole system for the entire evaluation period. The progress of training for several runs is plotted in Figure 5.

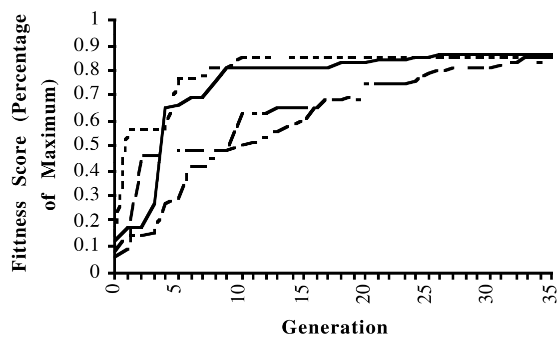

Figure 5. Progress of Training for Several Trials

A typical evolved rule base appears in the Appendix. Each row represents a single rule, and each column corresponds to a system variable. This system was found to effectively control each test case for more than one hour of simulated time. It was then applied to an initial configuration not included in the training set ($x = 0.85$, $\theta_a = 0.05$, and $\theta_b = -0.05$). Traces of these three variables as well as the forces recommended by the fuzzy system appear in Figure 6. The fuzzy controller was able to maintain the system in this configuration through more than 24 hours of simulated time. Since the physical system is inherently unstable, the fuzzy controller never actually brings cart and poles to rest, but rather establishes an oscillation which avoids system failure. In our example, the closest that the system came to being centered at rest was $x = 0.0090$, $\theta_a =$

-0.0001, θ_b = -0.0002, dx/dt = 0.0049, $d\theta_a/dt$ = 0.0037, and $d\theta_b/dt$ = 0.0040.

4. Discussion

The results obtained in this investigation compare favorably with those found in both [1] and [7]. A fuzzy system capable of controlling the jointed cart-pole system is constantly achieved within thirty generations. It is significant to note that even though this physical system requires two more variables than the single-pole system, the sizes of the trained bit strings do not increase significantly (as they would in [3], [4], and[6]), permitting the discovery of a solution in time comparable to the single-pole problem.

The fuzzy systems 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 genetic parameters, the training set, and the performance criteria. Currently, the genetic parameters are set by intuition, and trial and error—much like the way that the required number of hidden units in a neural network is determined. A detailed investigation of the relationship between the genetic parameters and the quality of evolved fuzzy controllers is currently underway. The composition of the training set is also critical to the performance of the evolved fuzzy controller, and must reflect both the range and distribution of the expected system input [2], [1]. Finally, the performance criteria must properly weight the selection pressures, otherwise a system may evolve which is consistent with the criteria but which does not perform as desired [2], [1].

The evolved fuzzy systems are also sensitive to the physical system parameters. Changing the length of either pole (initially 1.0m for the bottom pole and 0.1m for the top) by even a small amount increases the incidence of system failure. Future research will examine retraining previously evolved fuzzy systems with respect to a changed physical system. It may also be possible to evolve a more general fuzzy controller by training with respect to several pole configurations simultaneously.

Finally, the rule bases generated by this method are "black box" solutions. When considering how one would describe a solution for this problem, the system presented in the Appendix would certainly not be suggested. In this respect, these fuzzy systems are not unlike neural network-based controllers. However, in this case we have a way of "looking into" the black box and describing its operation by translating the fuzzy membership functions back into linguistic terms. Such an operation is typically not possible with neural networks.

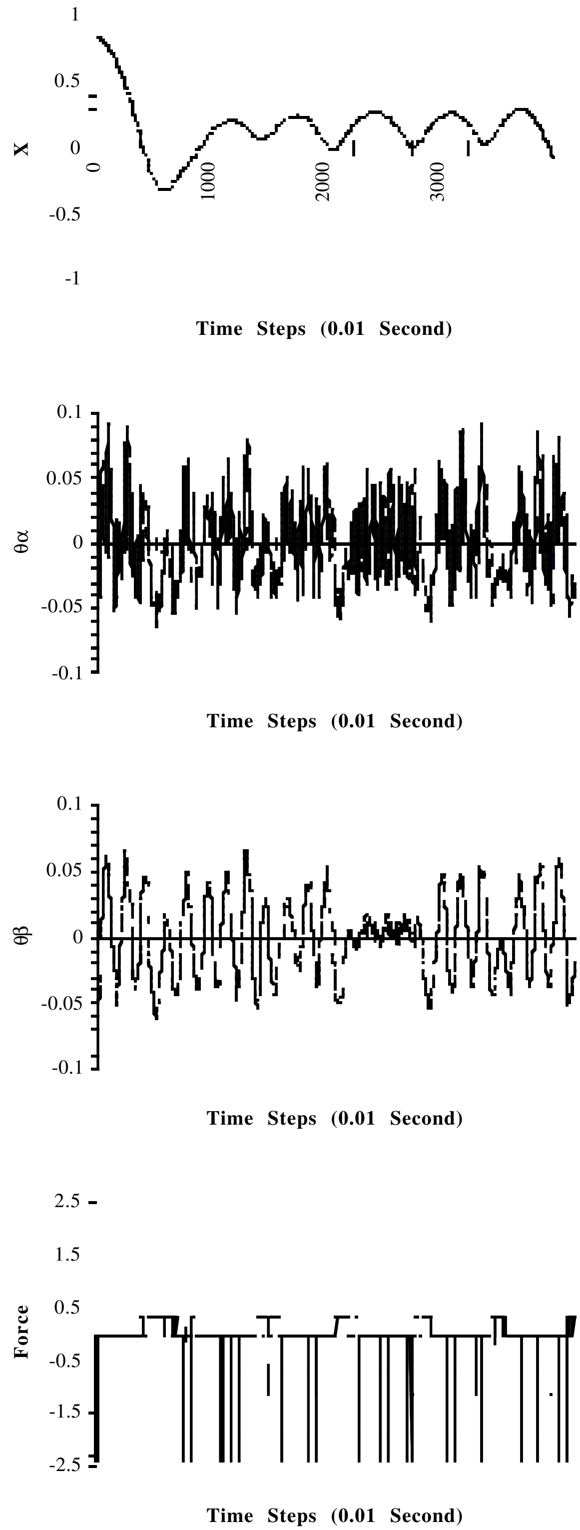

Figure 6. Traces of Selected Variables

With this paper we continue our research into the automatic generation of fuzzy controllers using genetic algorithms. We have shown the relative efficiency of using the compact rule base representation and have demonstrated its effectiveness on the cart and jointed-pole problem. We believe that this technique will be successful when applied to increasingly complex problems as we continue to characterize the methodology and explore its applications.

References

[1] Cooper, M.G. & J.J. Vidal (1993) "Genetic Design of Fuzzy Controllers", Presented at the Second International Conference on Fuzzy Theory and Technology, Durham, NC.

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

[3] Hu, Hernt-Tai, Heng-Ming Tai, & Sujeet Shenoi (1993) "Fuzzy Controller Design Using Genetic Algorithms and Cell Maps", Presented at the Second International Conference on Fuzzy Theory and Technology, Durham, NC.

[4] 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.

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

[6] 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.

[7] 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.

Appendix: Fuzzy Rule Base Produced by the Genetic Algorithm