Neural networks and genetic algorithm approaches to auto-design of fuzzy systems

Takagi, Hideyuki
Computer Science Division, University of California, Berkeley

LEE, Michael
Computer Science Division, University of California, Berkeley

Neural Networks and Genetic Algorithm Approaches to Auto-Design of Fuzzy Systems[0]

Hideyuki TAKAGI[1] and Michael LEE

Computer Science Division, University of California, Berkeley, CA 94720
takagi@cs.berkeley.edu, lee@cnmat.cnmat.berkeley.edu, FAX (510)642-5775

Abstract. This paper presents Neural Network and Genetic Algorithm approaches to fuzzy system design, which aims to shorten development time and increase system performance. An approach that uses neural network to represent multi-dimensional nonlinear membership functions and an approach to tune membership function parameters are given. A genetic algorithm approach that integrates and automates three fuzzy system design stages is also proposed.

1 Introduction

Fuzzy systems are frequently designed by hand. This poses two problems: (a) because hand design is time consuming, development costs can be very high; (b) there is no guarantee of obtaining an optimal solution. To shorten the development time and increase performance of fuzzy systems, there are two separate approaches: develop support tools and automatic design methods. The former includes developing environments to assist in fuzzy system design. Many environments are already commercially available. The latter approach involves introducing techniques to automate the design process. Though automatic design does not guarantee delivery of optimal solutions, they are preferable to manual techniques, because design is guided towards and an optimal solution by certain criteria.

There are three major design decisions to make when designing fuzzy systems:
(1) deciding the number of fuzzy rules,
(2) deciding the shape of the membership functions,
(3) deciding the consequent parameters.
Furthermore, two other decisions must be made:
(4) deciding the number of input variables,
(5) deciding the reasoning method.

(1) and (2) correspond to deciding how to cover the input space. They are highly dependent on each other. (3) corresponds to determining the coefficients

[0] This paper will appear in the Proc. of Fuzzy Logic in Artificial Intelligence (FLAI'93), Linz, Austria (June 28-30, 1993).
This research is supported in part by NASA Grant NCC-2-275, MICRO State Program Award No.90-191, and EPRI Agreement RP8010-34. We would like to thank Prof. David Wessel and the Center for New Music and Audio Technologies at UC Berkeley for use of computing resources.

[1] The author is a Visiting Industrial Fellow at UC Berkeley and a Senior Researcher of Central Research Laboratories, Matsushita Electric Industrial Co., Ltd.

of the linear equation in the case of the TSK (Takagi-Sugeno-Kang) model [1], or determining consequent part membership functions in the case of the Mamdani model [2]. (4) corresponds to deciding the minimum set of relevant input variables needed to compute target decisions or control values. Techniques, such as backward elimination [4] or information criteria are often used to do in this task. (5) corresponds to deciding which fuzzy operator and defuzzification method to use. Although several operators and fuzzy reasoning methods have been proposed, there is no criteria for selecting them. [5] shows that dynamically changing the reasoning method according to the reasoning environment results in higher performance and fault-tolerance than any one fixed reasoning method.

Neural networks (more generally, gradient based models) and genetic Neural networks (most commonly gradient based) and genetic algorithms are used for auto-design of fuzzy systems. The neural network based methods are mainly used to design membership functions. There are two major methods;
(a) Direct Multi-Dimensional Membership Functions Design:
This method first decides the number of rules by data clustering. Then the membership function shapes are learned by training on membership grades to each cluster. More detail will be given in section 2.
(b) Indirect Multi-Dimensional Membership Functions Design:
This method synthesizes multi-dimensional membership functions by combining one dimensional membership functions. The membership functions are tuned using gradient techniques which attempt to reduce the error between the desired output and actual output of the total fuzzy system.

The advantage of method (a) is that it can generate nonlinear multi-dimensional membership functions directly; there is no need to construct multi-dimensional membership functions by combining one dimensional membership functions. The advantage of method (b) is that it can be tuned by monitoring the final performance of the total fuzzy system. We review both methods in section 2.

Many of the methods that use genetic algorithms are similar in spirit to method (b); one dimensional membership functions shapes are automatically tuned using genetic algorithms. Many of these methods only consider one or two of the previously mentioned design issues. In section 3, we describe a method which decides design issues (1),(2), and (3) simultaneously.

2 Neural Network Approaches

2.1 Direct Fuzzy Partitioning of Multi-Dimensional Input Space

This approach uses neural networks to represent multi-dimensional nonlinear membership functions and is called NN-driven Fuzzy Reasoning [3, 4].

The advantage of this method is that it can generate nonlinear multi-dimensional membership functions directly. In conventional fuzzy systems, one dimensional membership functions used in the antecedent part, are independently designed and then combined to generate multi-dimensional membership functions indirectly. It can be argued that the neural network method is a more general form of the conventional fuzzy system in that the combination operations performed are absorbed by the neural network. Conventional indirect design methods have a problem when the input variables are dependent. For example, consider an air conditioner controlled by a fuzzy system that uses temperature and humidity as inputs. In conventional design methods of fuzzy systems, the membership functions of temperature and humidity are designed independently. The resulting fuzzy partitioning of the input space resembles Figure 1(a). However, when the input variables are dependent, such as temperature and humidity, fuzzy partitioning such as Figure 1(b) is more appropriate. It is very hard to construct such a nonlinear partitioning from one dimensional membership functions. Since NN-driven Fuzzy Reasoning constructs nonlinear multi-dimensional membership functions directly, it is possible to make the partitionings of Figure 1(b).

The design steps of NN-driven Fuzzy Reasoning had three steps: clustering the given training data, fuzzy partitioning the input space by neural networks, and designing the consequent part of each partitioned space.

The first step is to cluster the training data and decide the number of rules. Prior to this step, irrelevant input variables have already been eliminated using the backward elimination or information criteria methods. The backward elimination method arbitrarily eliminates one of the n input variables and trains the neural networks with $n-1$ input variables. The performance of neural networks with n and $n-1$ is then compared. If the performance of the $n-1$ input networks is similar or better than the n input networks, then the eliminated input variable is considered irrelevant. Next the training data is clustered and the distribution the data is obtained. The number of clusters is the number of rules.

The second step is to decide the cluster boundaries from the cluster information obtained in step 1; the input space is partitioned and the multi-dimensional input membership functions are decided. Supervised data is provided by the membership grade of input data to the cluster that is obtained in step 1. First a neural network with n inputs and c outputs, where n is the number of relevant input variables and c is the number of clusters determined in step 1, is prepared. Training data for this network, NN_{mem} in Figure 2, is generated by from the clustering information given by step 1. Generally, each input vector is assigned to one of the clusters. The cluster assignment is combined with the input vector to form a training pattern. For example, in the case of four clusters and an input vector which belongs to cluster 2, the supervised portion of the training pattern will be (0,1,0,0). In some cases, the user may intervene and manually construct the supervised portion if s/he believes an input data point should be classified differently than given by the clustering. For example, if the user believes that a data point belongs equally to class one and two, an appropriate supervised output pattern might be (0.5,0.5,0,0). After training this neural network on this training data, the neural network computes the degrees to which a given input vector belongs to each cluster. Therefore, we assume that this neural network acquires the characteristics of the membership functions for all rules by learning and can generate the membership value that corresponds to any arbitrary input vector. The fuzzy systems, which uses a neural network as the membership generator is the NN-driven Fuzzy Reasoning.

The third step is the design of the consequent parts. Since we know which cluster to assign an input data to, we can train the consequent parts using the input data and the desired output. A neural network expression can be used here as in [3, 4], but any other of the proposed methods, such as math equations or fuzzy variables, can be used instead. The essential point of this model is the neural networks which partition the input space in fuzzy clusters.

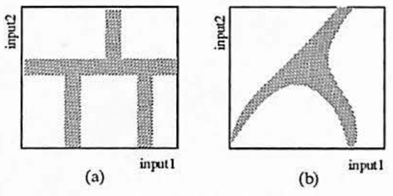

Fig. 1. Fuzzy partitioning: (a) conventional (b) desired

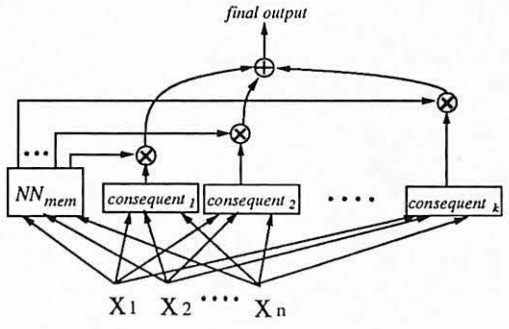

Fig. 2. Example structure of NN-driven Fuzzy Reasoning

Figure 2 shows one example of a NN-driven Fuzzy Reasoning system. This example is a model which outputs a singleton value computed by a neural network or a TSK model. Multiplication and addition in the figure calculate a weighted average. If the consequent part outputs fuzzy values, proper t-conorm and/or defuzzification operations should be used.

2.2 Tuning of Parameterized Fuzzy Systems

The parameters which define the shape of the membership functions are modified to reduce error between output of the fuzzy system and supervised data. Two methods which have been used to modify these parameters are: gradient-based methods and genetic algorithms. The genetic algorithm methods will be described in the next section and the gradient based methods will be explained in this section.

The procedure of the gradient based methods are: (1) decide how to parameterize the the shape of the membership functions (2) tune the parameters to minimize the actual output of the fuzzy system and the desired output using gradient methods, commonly steepest descent. Center position and width of the membership functions are commonly used shape definition parameters. Ichihashi et al. [6] and Nomura et al. [7, 8], Horikawa et al. [9][10], Ichihashi et al.[11] and Wang et al. [12], Jang [13][14] have used triangular, combination of sigmoidal, Gaussian, and bell shaped membership functions respectively. They tune the membership function parameters using steepest descent methods.

Figure 3 shows this method and is isomorphic to Figure 4. μ_{ij} in the figure is the membership function of input parameter x_j in the i-th rule, and actually it is represented by a parameter vector that describes the shape of the membership function. Namely, the method casts the fuzzy system as a neural network by

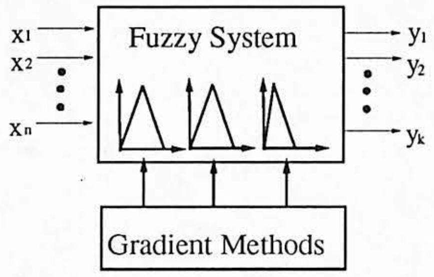

Fig. 3. Neural networks tune the parameters of fuzzy systems

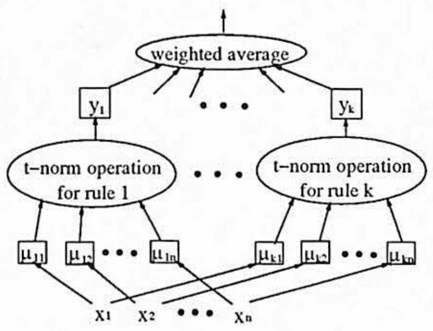

Fig. 4. Neural networks for tuning fuzzy systems: μ_{ij} is the membership function of input parameter x_j in i-th rule

representing membership functions as weights and rules by nodes which perform t-norm operations. Any network learning algorithm, such a backpropagation, can be used to train this structure. y_i in the figure is a trainable output value of each consequent part. This method has already been applied in the design of actual commercial products.

3 Genetic Algorithm Approaches

3.1 Genetic Algorithms and Fuzzy Control

Genetic algorithms are biologically inspired optimization techniques, which operate on populations of binary coded representations and perform reproduction, crossover, and mutation on them. The right to reproduce offspring for the next generation is based on a fitness value provided by the application. Genetic algorithms are attractive because they do not require the existence of derivatives, are robust, search many points simultaneously, and are able to avoid local minima.

Several papers have proposed automatic fuzzy system design methods using genetic algorithms. Much of the work has focused on tuning membership functions [17] - [25]. Other methods use genetic algorithms to determine the number of fuzzy rules [18, 26]. In [26], sets of rules are constructed by experts and the genetic algorithm finds the best combination of them. In [18], Karr has developed a method for determining membership functions and number of fuzzy rules. In this paper, Karr's method first uses a genetic algorithm to determine the number of rules according to a predefined rule base. Following this stage, the method uses a genetic algorithm to tune the membership functions. Although these methods produce systems that perform better than human designed systems, they may be suboptimal because they treat only one or two of the three major design stages at a time. Because these design stages may not be independent, it is important to consider them simultaneously to find the optimal solution. In the next section, we propose an automatic design method that integrates the three major design stages.

3.2 Integrated Design of Fuzzy Systems using Genetic Algorithms

This section proposes an automatic fuzzy system design method that uses a genetic algorithm and integrates the three major design stages: the membership function shapes, the number of fuzzy rules, and the rule-consequent parameters are determined at the same time [27].

There are two major steps to perform when applying genetic algorithms to an application; (a) to choose a suitable genetic representation, and (b) to design an evaluation function to rank members of the population. In the following paragraphs, we discuss our fuzzy system representation and genetic representation. An evaluation function and methods for embedding apriori knowledge will be presented in the next section.

Fuzzy System and Genetic Representations We use the TSK model fuzzy system, which is widely used in control problems, to map the state of the system to a control value. The TSK model differs from conventional fuzzy systems in that the consequent parts of a TSK fuzzy model are expressed by linear equations as opposed to fuzzy linguistic expressions. For example, a rule in a TSK model has the form:

IF X_1 is A and X_2 is B THEN $y = w_1X_1 + w_2X_2 + w_3$,

where w_n are constants. The final control value is computed by summing the output of each rule and weighting it according to the rule's firing strength.

For the antecedent part, we use triangular membership functions parameterized by left base, right base, and distance from the previous center point (the first center is given as an absolute position). Other parameterized shapes, such as sigmoidal, Gaussian, bell, or trapezoidal can be substituted. Unlike most methods, overlap restrictions are not placed on the sets in our system and the possibility of complete overlap exists (see Figure 5).

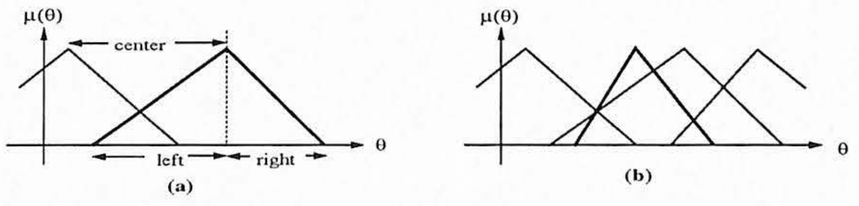

Fig. 5. (a) membership function representation (b) possible member functions

Generally, the number of fuzzy sets for each input variable combine to determine the number of fuzzy rules. For example, a TSK fuzzy system with m input variables, each partitioned into n fuzzy sets, would yield n^m fuzzy rules. Because the number of rules depends directly on the number of membership functions, eliminating membership functions has the direct effect of eliminating rules.

Each membership function requires three parameters and each fuzzy rule requires three parameters. Thus, an m-input-one-output system with n fuzzy sets per input variable requires $3(mn + n^m)$ parameters.

The genetic representation explicitly contains the three membership function parameters and the consequent parameters as mentioned previously. The number of rules, however, are encoded implicitly via the application boundary conditions and membership function positions. We can implicitly control the number of rules by eliminating membership functions whose center positions lie outside the range of its corresponding input variable and rules which contain them. For example, in the inverted pendulum application, rules using θ membership functions with center positions greater than 90° can be eliminated. This implies that the number of rules can be optimized at the same time the genetic algorithm optimizes the shape of the membership functions and the consequent parameters.

The actual genetic code is formulated by defining a chromosome as a set of parameters that represent a higher level entity, such as a membership function or rule-consequent parameter set (see Figure 6). Figure 7 shows chromosomes linked together to form the entire fuzzy system representation.

center	left base	right base
10100110	10011000	01011000

membership function chromosome (MFC)

w_1	w_2	w_3
10100110	10011000	01011000

rule-consequent parameters chromosome (RPC)

Fig. 6. Composite chromosomes

fuzzy variable θ			fuzzy variable $\delta\theta/\delta\tau$			rule-consequent parameters		
MFC_1	$\cdots$	MFC_{10}	MFC_1	$\cdots$	MFC_{10}	RPC_1	$\cdots$	RCP_{100}

Fig. 7. Gene map

3.3 Evaluation

Evaluation of Inverted Pendulum Controllers The inverted pendulum represents a classic non-linear control problem where the task is to find a control strategy that can balance a pole on a movable cart. In our simulation, the movement of both the pole and the cart is restricted to the vertical plane and the cart is allowed to move infinitely in either direction along the track. The controller uses the angular displacement and velocity to compute a force, which is applied throughout the control interval. More details can be found in [14] and [27].

Unlike the fuzzy system representation, the evaluation function relies directly on the application. The genetic algorithm uses an objectively computed performance measure to guide the search for better and better controllers. An inverted pendulum trial can end in three different conditions: (1) the pole becomes balanced, (2) simulation time runs out, or (3) the pole falls over. t_{end} represents the time when the trial actually ends and t_{max} represents the maximum simulation time. According to these three conditions, we came up with the following guidelines: if the system balances the pole, a shorter time is better than a longer time;

if the pole falls over, a longer time until failure is better than a shorter time. Figure 8 captures this notion.

$$score(t_{end}) = \begin{cases} a_1\,(t_{max} - t_{end}) + a_2 reward & (1) \\ reward & (2) \\ b \cdot t_{end} & (3) \end{cases}$$

where a_1, a_2, b and $reward$ are constants, and (1) pole balanced, (2) $t_{max} = t_{end}$, and (3) pole fell over ($|\theta| \geq 90°$). t_{end} represents the time when 525 the trial actually ends and t_{max} represents the maximum 526 simulation time.

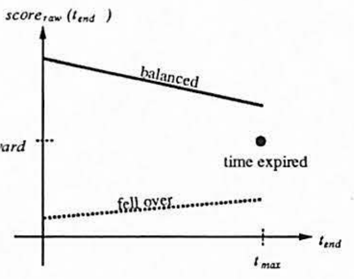

Fig. 8. Raw scoring functions

It is also desirable to have the controller work well over a wide range of initial conditions. To consider this point, we evaluated a controller by summing the scores obtained from individual trials performed on a set of initial conditions. We then augmented our evaluation function with additional terms to consider steady state error and to penalize systems according to the number of rules in the system. The resulting fitness score for one trial was computed as

$$score(t_{end}) = \frac{\left(score_{raw}(t_{end}) + c \sum_0^{t_{end}} |\theta_t|\right)}{\text{number of rules} + \text{offset}_{\text{rules}}}$$

The steady state error was a weighted integral of the angle displacement and the $\text{offset}_{\text{rules}}$ parameter controlled the degree of penalty for number of rules.

Embedded knowledge There are two levels at which apriori knowledge can be added to our fuzzy system design method: at the level of fuzzy systems and at the level of the application. We can incorporate our knowledge of fuzzy systems and inverted pendulums into our system in several ways: via initial conditions, via the representation of the fuzzy system, or via the objective function [28].

Traditional genetic algorithms start with randomly generated population of solutions. The purpose of this procedure is to randomly cover the solution space with the hope that one may be near the optimal solution. We can use our application knowledge to initialize some members of the population with good parameter sets. Biasing a population in this manner does not restrict the genetic algorithm to searching for solutions in the neighborhood of the initial solutions, because the mutation, crossover, and reproduction operations still allow the genetic algorithm to search points far from the initial points. If the initial solutions are approximately correct, we can gain significant speedup. However if they are near a local minimum, the genetic algorithm can be temporarily distracted and require more time to find the optimum.

From our experience with fuzzy systems, we know that equally partitioning the input dimensions is a good idea. Using this knowledge, we can initialize a few members of our population to cover the input dimensions in this manner. Our initial population includes members that equally divide the input dimension into varying number of fuzzy sets in addition to randomly generated members.

The inverted pendulum has been well studied and control laws such as

$$force = c_1 sin(\theta) + c_2 \frac{\delta\theta}{\delta t}$$

have been developed. We can approximate this function by setting the constant parameter, w_3, of the TSK consequent part to the value of the control law when it is evaluated at the center point of the antecedent membership functions.

We can also take advantage of the symmetrical nature of the inverted pendulum balancing task by requiring the underlying fuzzy system to be symmetric. An example symmetric fuzzy system would symmetrically partition the input space about the origin and constrain the consequent parameters to reflect symmetry. In addition to possibly improving the performance of the resulting system, the number of initial conditions to train upon can be reduced by half.

An alternative technique to incorporate symmetry into the design system is to implicitly include it into the evaluation function. By including symmetric initial starting points in the evaluation process, we can insure that symmetric conditions are considered.

3.4 Experimental Method and Results

Our method combines a genetic algorithm, a penalty strategy, and unconstrained membership function overlap to automatically design fuzzy systems. In this section, we first present results which do not make use of any apriori knowledge. We then compare these results with results obtained from our method where apriori knowledge has been incorporated into the design process.

Experimental Method There are experimental parameters associated with the genetic algorithm and parameters associated with the application. Crossover rate, number of crossover points, mutation rate, population size, and number of generations to produce are parameters of the genetic algorithm. The maximum fuzzy sets per input variable and parameter bit resolution are parameters associated with the fuzzy system and fuzzy system coding. The $\text{offset}_{\text{rules}}$, maximum simulation time, reward, and balancing criteria are parameters used for evaluating controllers.

We used a genetic algorithm with two point crossover and set crossover rate to 0.6, the mutation rate to 0.0333, and the population size to 10. The number of generations produced depended on the experiment. We used an elitist strategy in which the member with the highest fitness value automatically advanced to the next generation.

The maximum fuzzy sets per input variable was set to ten in our experiments. This limit was set through experience with the inverted pendulum application. Because we included a penalty strategy that involves the number of rules, the setting of this number is not so critical. The precision of all parameters was set to 8 bits. The resulting genetic representation for fuzzy systems used in our experiments consisted of 360 parameters or 2880 bits (see Figure 6).

The balancing criteria was set to 0.0001 in our experiments. The pole is considered balanced if the following condition is met:

$$(\theta(t) - \theta(t-1))^2 + (\dot{\theta}(t) - \dot{\theta}(t-1))^2 < \text{balance criteria}.$$

The reward was set to 5000 and the maximum simulation time t_{end} was 200. The evaluation parameter $\text{offset}_{\text{rules}}$ was set to 10.

Experimental Results After all of these parameters have been set, the genetic algorithm begins its search. Each controller in the population was evaluated with the set of initial conditions: $(\theta, \delta\theta/\delta t)$ = (5.22, 6.93), (5.11,6.97), (-8.41, -1.37), (6.22, -7.14). After completing the requested number of iterations, the best solution is kept and the rest are discarded. In one experiment, the method produced a system with only four rules. Their symmetric rules are:

$$\text{IF } \theta \text{ is } A_i \text{ and } \dot{\theta} \text{ is } B_i, \text{ THEN } y = w_{1i}\theta + w_{2i}\dot{\theta} + w_{3i},$$

where $i = 1 \sim 4$. The obtained parameters in four consequent parts were (w_{1i}, w_{2i}, w_{3i}) = (0.44, 1.02, -31.65), (1.54, -0.61, -30.14), (1.54, -0.61, 30.14), and (0.44, 1.02, 31.65). The obtained triangular membership functions, A_i and B_i were $A_1 = A_3$ = {-119.65, -62.12, 4.59}, $A_2 = A_4$ = {-4.59, 62.12, 119.65}, $B_1 = B_3$ = {-219, -1.99, 238.56}, and $B_2 = B_4$ = {-238.56, 1.99, 219.64}. Figures 9 and 10 show trajectory plots for several initial conditions.

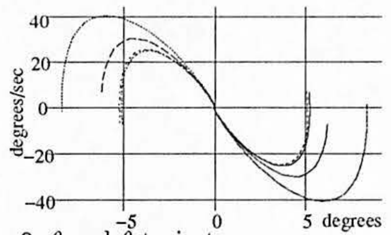

Fig. 9. θ and $\dot{\theta}$ trajectory

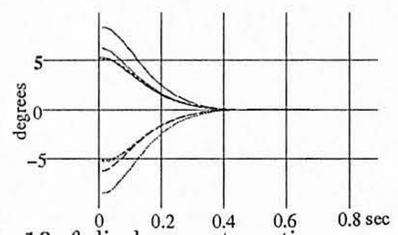

Fig. 10. θ displacement vs. time

We have designed four experiments to study the effect of including different combinations of apriori knowledge on the performance of both the design method and the resulting systems. The forms of apriori knowledge embedded into the design method were symmetric fuzzy system structure and heuristic initialization. The fitness vs generation plot appears in Figure 11 with an experimental condition table.

Each experiment performed 5000 iterations on eight symmetric initial conditions. The experiments that used symmetric fuzzy systems used only half of the points and thus performed half as many function evaluations as the asymmetric systems. It is interesting to note that while the experiments heuristically initialized initially had superior performance than those randomly initialized, they required more time to obtain higher fitness levels. This may be due to the fact that the initial conditions given represented a local minimum.

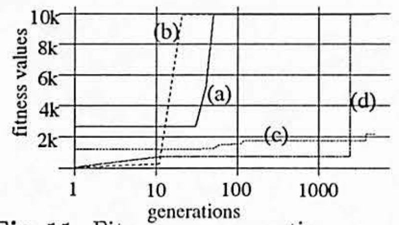

Fig. 11. Fitness vs. generation

Fuzzy rules and initialization of genetic algorithm parameters

experiment	Fuzzy System	Initialization
(a)	symmetry rules	heuristic
(b)	symmetry rules	random
(c)	asymmetry rules	heuristic
(d)	asymmetry rules	random

4 Conclusion

Neural networks and genetic algorithms have matured into practical tools for automatic fuzzy system design. Neural network approaches for automatically and completely specifying fuzzy systems have already made their commercial debut. Neural networks and genetic algorithms for automatic fuzzy system design are just two examples of using one technology to strengthen another. Many more combinations involving two or more cooperating technologies are waiting to be discovered and explored.

References

1. Takagi, T. and Sugeno, M., "Fuzzy Identification of Systems and Its Applications to Modeling and Control," IEEE Trans. SMC-15-1, 1985, pp.116-132
2. Mamdani, E. H., "Applications of fuzzy algorithms for control of simple dynamic plant," Proc. of IEEE, Vol. 121, No. 12, pp.1585-1588 (1974)
3. Takagi, H. and Hayashi, I., "Artificial_neural_network-driven fuzzy reasoning," Int'l Workshop on Fuzzy System Applications, pp.217-218 (Aug., 1988)
4. Takagi, H. and Hayashi, I., "NN-driven Fuzzy Reasoning," Int'l J. of approximate Reasoning, Vol. 5, No.3, pp.191-212 (1991)
5. Smith, M.H., "Parallel Dynamic Switching of Reasoning Methods in a Fuzzy System," 2nd IEEE Int'l Conf. on Fuzzy Systems, vol.2, pp.968-973 (March, 1993)
6. Ichihashi, H. and Watanabe, T., "Learning Control by Fuzzy Models Using a Simplified Fuzzy Reasoning", J. of Japan Society for Fuzzy Theory and Systems. Vol. 2, No.3, pp.429-437 (1990), (*in Japanese*)
7. Nomura, H. Hayashi, I. and Wakami, N., "A self-tuning method of fuzzy control by descent method," 4th IFSA World Congress, Vol. Engineering, pp.155-158 (July, 1991)
8. Nomura, H. Hayashi, I. and Wakami, N., "A learning method of fuzzy inference rules by descent method," IEEE Int'l Conf. on Fuzzy System, pp.203-210 (March, 1992)
9. Horikawa, S., Furuhashi, T., Okuma, S., and Uchikawa, Y., "Composition Methods of Fuzzy Neural Networks," Int'l Conf. on Ind., Elect., Control, Instr., and Automation , pp.1253-1258 (Nov., 1990)
10. Horikawa, S., Furuhashi, T., and Uchikawa, Y. "On Fuzzy Modeling Using Fuzzy Neural Networks with the Back-Propagation Algorithm," IEEE Trans. Neural Networks. Vol.3, No.5, pp.801-806 (1992)
11. Ichihashi H. and Tanaka, "Backpropagation Error Learning in Hierarchical Fuzzy Models," Symposium of SICE Kansai Chapter, pp.131-136 (1990) (*in Japanese*)

12. Wang, L-X. and Mendel, J.M., "Back-Propagation Fuzzy System as Nonlinear Dynamic System Identifier," IEEE Int'l Conf. on Fuzzy Systems, pp.1409-1418 (March, 1992)
13. Jang, J-S. "Rule Extraction Using Generalized Neural Networks," 4th IFSA World Congress. Vol.Artificial_Intelligent, pp.82-86 (July, 1991)
14. Jang, J-S. "Self-Learning Fuzzy Controllers Based on Temporal Back Propagation," IEEE Trans. Neural Networks. Vol.3, No.5, pp.714-723 (1992)
15. Takagi, H., "Cooperative system of neural networks and fuzzy logic and its application to consumer products," (edited by J. Yen and R. Langari) Industrial Applications of Fuzzy Control and Intelligent Systems, Van Nostrand Reinhold (will be published in 1993)
16. Asakawa, K. and Takagi, H., "Neural Networks Applications in Japan," Communications of ACM, (submitted)
17. Karr, C., Freeman, L., Meredith, D., "Improved Fuzzy Process Control of Spacecraft Autonomous Rendezvous Using a Genetic Algorithm," SPIE Conf. on Intelligent Control and Adaptive Systems, pp.274-283 (Nov., 1989)
18. Karr, C., "Applying Genetics to Fuzzy Logic," AI Expert, Vol.6, No.2, pp.26-33 (1991)
19. Karr, C., "Design of an Adaptive Fuzzy Logic Controller using a Genetic Algorithm," Int'l Conf. of Genetic Algorithms, pp.450-457 (July, 1991)
20. Karr, C., and Gentry, E., "A Genetics-Based Adaptive pH Fuzzy Logic Controller," Int'l Fuzzy Systems and Intelligent Control Conf., pp.255-264 (March, 1992)
21. Karr, C., Sharma, S., Hatcher, W., and Harper, T., "Control of an Exothermic Chemical Reaction using Fuzzy Logic and Genetic Algorithms," Int'l Fuzzy Systems and Intelligent Control Conf., pp.246-254 (March, 1992)
22. Nishiyama, T., Takagi, T., Yager, R., and Nakanishi, S., "Automatic Generation of Fuzzy Inference Rules by Genetic Algorithm," 8th Fuzzy System Symposium, pp.237-240 (May, 1992) (*in Japanese*)
23. Nomura, H., Hayashi, I., and Wakami, N., "A Self-Tuning Method of Fuzzy Reasoning By Genetic Algorithm," Int'l Fuzzy Systems and Intelligent Control Conf., pp.236-245 (March, 1992)
24. Qian, Y., Tessier, P., and Dumont, G., "Fuzzy Logic Based Modeling and Optimization," 2nd Int'l. Conf. on Fuzzy Logic and Neural Networks , pp.349-352 (July, 1992)
25. Tsuchiya, T., Matsubara, Y., and Nagamachi, M., "A Learning Fuzzy Rule Parameters Using Genetic Algorithm," 8th Fuzzy System Symposium, pp.245-248 (March, 1992) (*in Japanese*)
26. Takahama, T., Miyamoto, S., Ogura, H., and Nakamura, M., "Acquisition of Fuzzy Control Rules by Genetic Algorithm," 8th Fuzzy System Symposium, pp.241-244 (March, 1992) (*in Japanese*)
27. Lee, M., and Takagi, H. "Integrating Design Stages of Fuzzy Systems using Genetic Algorithms," 2nd IEEE Int'l Conf. on Fuzzy Systems, vol.1, pp.612-617 (March, 1993)
28. Lee, M., and Takagi, H. "Embedding Apriori Knowledge into an Integrated Fuzzy System Design Method Based on Genetic Algorithms," 5th IFSA World Congress (1993) (to appear)
29. Goldberg, D., "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley (1989)

This article was processed using the LaTeX macro package with LLNCS style