GENETIC ALGORITHMS for FUZZY CONTROLLERS

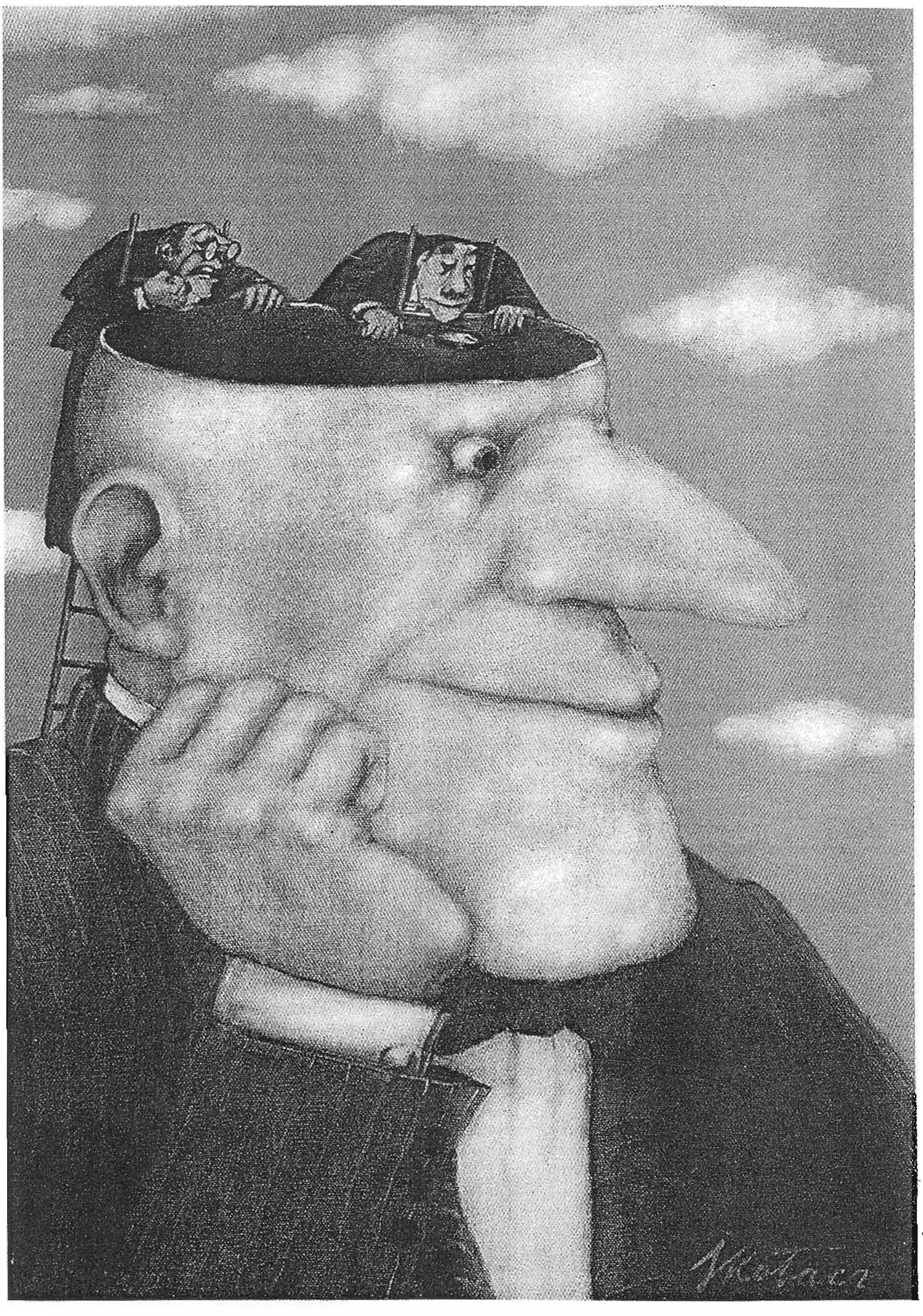
By Chuck Karr

The performance of expert systems in process control is limited by their inflexible representation of human decision making. This inflexibility can be combatted by incorporating fuzzy set theory, in which abstract or subjective concepts can be represented with linguistic variables such as *high* and *low*. These variables are used in rule sets to control physical systems.

Fuzzy-logic controllers (FLCs) are rule-based systems that use fuzzy linguistic variables to model human rule-of-thumb approaches to problem solving; they have been successful for several control problems. These "fuzzy expert systems" feature rules that direct the decision process, and membership functions that convert linguistic variables into the precise numeric values needed by the computer. The rule set is gleaned from a human expert's knowledge, which is based on experience, and can usually be written more easily than the crisp rule sets required by most conventional expert systems.

Defining fuzzy membership functions, however, is almost always the most time consuming aspect of FLC design, and the problem isn't getting any easier. This chore is often painstakingly accomplished by trial and error: unfortunately, one single change in the membership functions can significantly alter the controller's performance.

T. J. Procyk and E. H. Mamdani introduced an iterative procedure for altering membership functions but, in general, the development of methods for choosing functions that optimize FLC performance has received little attention. A standard method for determining the membership functions that produce maximum FLC performance is



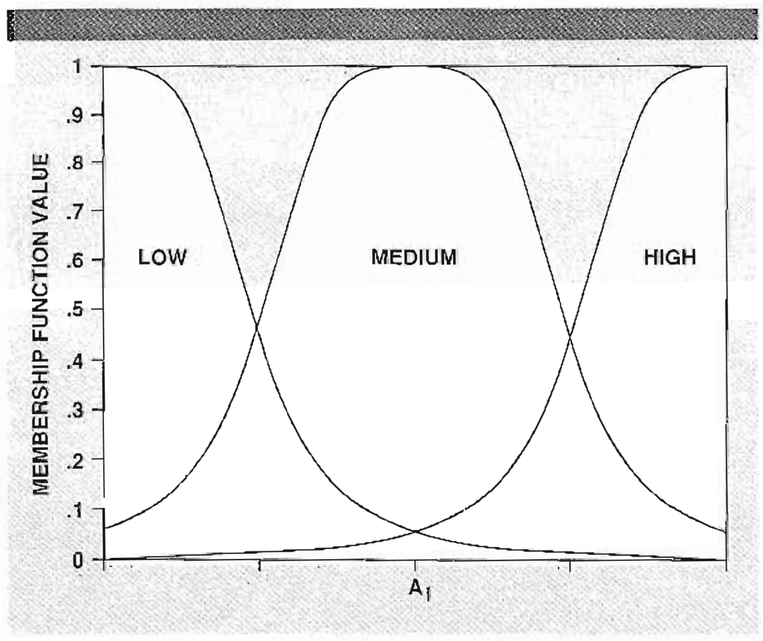


FIGURE 1. Fuzzy membership functions define the linguistic variables.

needed, but this goal poses a substantial search problem because a high degree of nonlinearity can appear in the search.

Genetic algorithms (GAs) are a search technique drawing increasing attention in the field of optimization. GAs are algorithms that use operations found in natural genetics to guide their trek through a search space. R. B. Hollstien and K. A. De Jong have demonstrated the technique's efficiency in function optimization. De Jong's work, in particular, establishes the GA as a robust search technique efficient across a spectrum of problems, as compared to other traditional schemes. Subsequent application of GAs to the search problems of pipeline engineering, VLSI microchip layout, structural optimization, job-shop scheduling, equipment design, and machine learning support the claim that GAs are broadly based.

The robust nature and simple mechanics of GAs make them inviting tools not only for establishing membership functions, but also for selecting rules to be used in FLCs. GAs are also potentially useful for selecting rules due to some special traits that differentiate them from conventional techniques.

FLC MECHANICS

Several approaches to FLC development are possible. Generally, a compositional "rule of inference" (a mathematical statement describing how the linguistic variables are to be manipulated) is employed to control the problem environment. However, this rule is often described by complex fuzzy mathematics foreign to anyone unfamiliar with fuzzy logic. Instead, I'll present a hands-on, rational approach to FLC development used at the U.S. Bureau of Mines and describe the procedure in generic form.

The first step is to determine which variables will be important in choosing an effective control action. Any number of these decision variables may appear, but the more that are used, the larger the rule set that must be written. (Decision variables are simply the variables upon which the control decisions are based.)

Once the decision variables have been chosen, the control variables must be identified. Unlike decision variables, the number of control variables has no impact on the size of the rule set—they are the "knobs" that can be turned to effect a change on the system.

Next, the fuzzy terms that will be used to describe both the decision and control variables must be defined (these terms are sometimes called fuzzy sets). The variables allow the FLC to model the human decision-making process; you should choose terms similar to those used by human controllers. Fuzzy terms allow for the most straightforward production of a rule set.

The choice of fuzzy sets defines the number of fuzzy rules required; a fuzzy rule is written for every possible condition that could exist in the physical system. The rules, commonly called production rules, are of the form:

IF [c_1 is (C_1) AND c_2 is (C_2) AND ...]
$\rightarrow$ THEN [a_1 is (A_1) AND a_2 is (A_2) AND ...]

where C_i is fuzzy sets characterizing the respective decision variables, and A_i is fuzzy sets characterizing the control variables. Although writing rules for all possible conditions in the physical system seems imposing at first, the incorporation of fuzzy terms into the rules makes their development much easier than that of conventional expert-system rules.

FLCs are based on the idea that some uncertainty exists in categorizing the values of the variables: fuzzy sets mean different things to different people. As a result, some mechanism for interpreting the fuzzy sets is necessary. The fuzzy membership function serves this purpose: it allows precise numeric values of the decision variables to be transformed into fuzzy sets and the fuzzy-control actions of the production rules to be transformed into precise, discrete control actions.

Fuzzy membership functions can be thought of as approximations of the confidence with which a precise numeric value is described by a fuzzy set, and fuzzy membership function values (f) are numeric representations of these confidences. When a fuzzy membership function has a value of $f = 1$, a maximum confidence exists that the precise numeric value is accurately described by the fuzzy set (Figure 1). For every precise decision value, each fuzzy set has a membership function value. For instance, a decision variable is described by the fuzzy terms *low*, *medium*, and *high* simultaneously with varying degrees of confidence.

Now that the precise numeric conditions in the system at any given time can be categorized in a fuzzy set with some certainty, a process for determining a precise value of the actions to be applied must be developed. This task involves writing a rule set that provides a fuzzy action (one of the fuzzy sets chosen to describe the control variables) for any condition that could possibly exist in the problem environment. Therefore, a human expert provides a fuzzy action for each condition possible in the environment. (The formation of the rule set is comparable to that of an expert system, except that the rules incorporate linguistic variables with which human operators are comfortable.) The use of fuzzy sets allows rules to be derived based mostly on the expert's intuition, although some experience with the system to be controlled certainly makes the development of the rule set more efficient.

At this point, a means for converting a precise set of conditions to a set of fuzzy conditions, and a set of fuzzy rules prescribing a fuzzy action associated with a particular set of fuzzy conditions, have been developed. We still have to convert the individual fuzzy actions provided by the fuzzy rules into a single, precise set of actions to be taken on the physical system.

L. I. Larkin found that a procedure known as the center-of-area scheme is an efficient method for determining these precise actions. In this method, the fuzzy membership functions describing the control actions are used in a weighted summing procedure to find one precise action. The scheme for finding the weighted sum can be easily thought of in graphical terms: each production rule has its action membership function plotted with a height equal to the minimum confidence (degree of membership) associated with the condition portion of the rule (Figure 2). The single value of the control actions, associated with the centroid, is the single value of the action to be applied to the physical system. The rules in which one has the greatest confidence produce shapes with the largest areas and thus have the greatest effect on the selection of the action. This process is summarized in this pseudocode, which describes the implementation of the center-of-area method for a set of triangular membership functions:

```
for (first rule to last rule) begin.
 GetMuVals /* fuzzify the conditions in physical
            system */
 if (all Mu values for this rule > 0.0) begin
 TriagArea = (Base * MuMin) / 2
 TriagMidpt = BasePosition + (BaseLength / 2)
 SumArea = SumArea + TriagArea
 SumMidpt = SumMidpt + TriagMidpt
 end end Action = SumArea / SumMidpt
```

Producing an FLC that controls a particular system is reasonably easy. However, to produce an efficient controller, the FLC must be properly designed. The rules must adequately model the human's approach to controlling the system, and the membership functions must be massaged until the controller performs acceptably for the spectrum of conditions that could exist in the controlled environment.

GENETIC ALGORITHMS

GAs have some properties that make them inviting as a technique for selecting high-performance membership functions for FLCs. Due to these properties, GAs differ fundamentally from more conventional search techniques. For example, they:

- Consider a population of points, not a single point
- Work directly with strings of characters representing the parameter set, not the parameters themselves
- Use probabilistic rules to guide their search, not deterministic rules.

GAs consider many points from the search space simultaneously and therefore have a reduced chance of converging to local optima. These populations are generated and tested iteration by iteration. This process is similar to a natural population of biological creatures in which successive generations of organisms are produced and raised until they themselves are ready to reproduce. In most conventional search techniques, a point-by-point search is conducted wherein a single point is selected, tested, and used with some decision rule to select the next point to be considered.

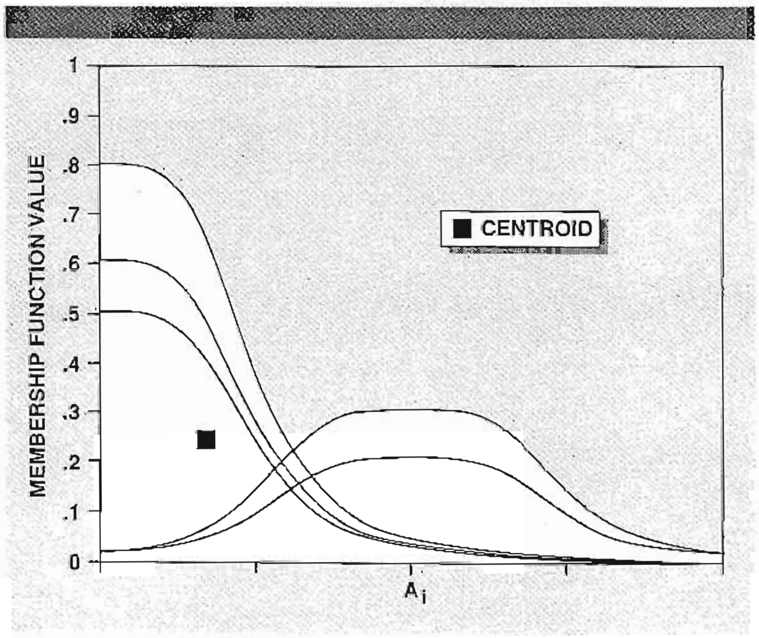


FIGURE 2. The center-of-area method is used to determine a single action.

Which expert system shell also has HyperWindows?

The New CxPERT 3.0

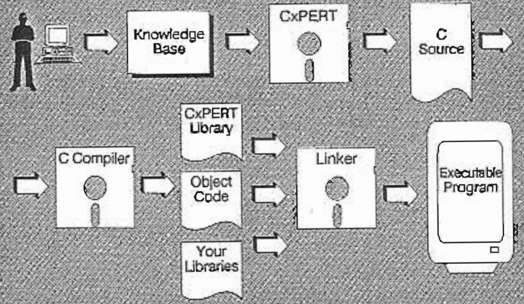
Get out of the old shell game and pick the new CxPERT 3.0... the expert system shell and hypertext tool that's versatile, simple-to-use, and lets you get right to work developing and *delivering royalty-free applications.*

The open architecture & unlimited versatility of CxPERT let you:

- Mix C code with the CxPERT representation language
- Translate knowledge bases directly into C
- Embed knowledge bases in C applications
- Deliver fast, royalty-free applications

And now, with CxPERT's new HyperWindows interface and text display system, you can easily add the power of hypertext to *any* C application.

Here's how CxPERT 3.0 works:



So remember, it doesn't take luck, it takes programming smarts to pick CxPERT 3.0! Priced at just **$795** (PC version), it's a sure bet... give it a crack. ***Send for* FREE *Info Kit. Demo Disk $10. Call (301) 261-0264.***

Software Plus

1652 Albermarle Drive ▪ Crofton, MD 21114

CIRCLE 15 ON READER SERVICE CARD

These methods can be dangerous in multimodal (many peaked) search spaces because they can converge to local optima. However, GAs generate entire populations of points (coded strings), test each point independently, and combine qualities from existing points to form a new population containing improved points. Aside from producing a more global search, the GA's simultaneous consideration of many points makes it highly adaptable to parallel processors since the evaluation of each point is an independent process. Avoiding convergence to a local optimum is important in selecting high-efficiency membership functions because the search spaces are often poorly behaved. Compatability with parallel processors is important in the selection of rules for control systems because adaptive FLCs—those in which the rule sets change during the process—put a premium on execution speed.

GAs require the natural parameter set of the problem to be coded as a finite length string of characters. It has become standard practice to use bit strings to represent the possible solutions to search problems. In some problems, the creation of appropriate finite string codings may require complicated mappings, but with a little creativity the possibilities are endless. Fortunately, GAs have been used long enough for the development of some fairly standard codings. Because they work directly with a coding of the parameter set and not the parameters themselves, GAs are difficult to fool because they're not dependent on continuity of parameter space. This characteristic is certainly important in the design of FLCs, because the meaning of the term "continuity of the search space" isn't even clear.

A GA requires only information concerning the quality of the solution produced by each parameter set (objective function value information). This characteristic differs from optimization methods that require derivative information or, worse yet, complete knowledge of the problem structure and parameters. (Imagine trying to produce derivative information for different rules and membership function definitions in an FLC.) Since GAs do not require such problem-specific information, they're more flexible than most search methods.

GAs also differ from other search techniques in that they use random choice to guide their search. Although chance is used to define decision rules, GAs are not random walks through the search space. They use random choice efficiently in their exploitation of prior knowledge to locate near-optimal solutions rapidly.

SIMPLE GENETIC ALGORITHM

A simple GA that has given good results in a variety of engineering problems comprises three operators: reproduction, crossover, and mutation. These operators are implemented by performing the basic tasks of copying strings, exchanging portions of strings, and generating random numbers—those easily performed on a computer. Before looking at the operators, consider the overall processing of a GA during a single generation. It begins by randomly generating a population of N strings, each of length m.

Remember that each string represents one possible solution to the problem: one set of fuzzy membership functions if searching for high-efficiency membership functions, or one set of actions associated with a set of conditions if searching for a rule set. Each of these strings is decoded so the character strings yield the actual parameters.

The parameters are sent to some conceptual framework that yields a measure of the solution's quality (often a mathematical model of the physical system is used), evaluated with some objective function (told how good an FLC the parameters produce), and assigned a fitness value that is simply a non-negative measure of relative worth. Some GA reproduction operators do not require the fitness values to be non-negative.

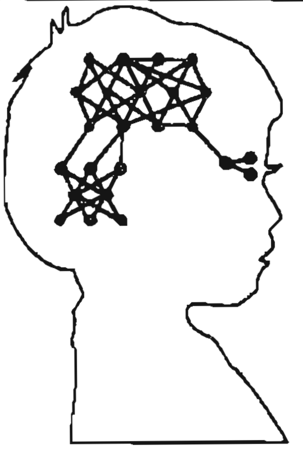
The fitness is used when employing the three operators that produce a new population of strings (a new generation). With luck, this new generation will contain better solutions to the problem. Even if improved solutions are not found on the microscopic scale from one particular generation to the next, GAs will locate improved solutions on the macroscopic scale over several generations. The new strings produced in subsequent generations are again decoded, evaluated, and transformed using the basic operators. The process continues until convergence is achieved or a suitable solution is found.

Reproduction is a process by which strings with large fitness values (good solutions to the problem at hand) receive correspondingly large numbers of copies in the new population. For example, in roulette-wheel selection, those strings with high fitness values f_i are given a proportionately higher probability of reproduction selection, p_{select_i}, according to this distribution:

$$p_{select_i} = \frac{f_i}{\Sigma f}$$

Once the strings are reproduced or copied for possible use in the next generation, they are placed in a mating pool where they await the action of the other two operators.

In elitist reproduction, the survival of the current best string is ensured from generation to generation by allowing one copy of the best string to pass undaunted into the next generation. This approach guarantees



"BrainMaker is the most fascinating computer software I've ever seen."
John Dvorak
PC Magazine

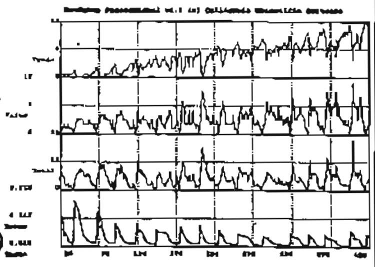
New! BrainMaker V2.2

The #1 selling Neural Network

Over 9,000 copies sold. Here's why:

Best documentation: 422 pg Users Guide, 255 pg *Introduction to Neural Networks* - everything you need to build networks and applications now.
Easiest to use: Pull down menus and dialog boxes. Supports mouse and color. Reads your data from dBase, Lotus, Excel, and ASCII & binary.

Fastest:
128,000 cps on 286/8
250,000 cps on 286/20
500,000 cps on 386/33
3 Million cps on C25
10 M cps on MD1220
2 Billion cps on i80170



Most complete product line:
- **BrainMaker** including User's Guide and *Intro*. 512 neurons per layer up to 8 layers: **only $195**
- **NetPlotter** graphics post-processor: **$79.95**
- **Hypersonic Training Program (HTP)** will train your network in minutes: **$149.95**
- **BrainMaker Professional** including graphics, run-time license with source code, HTP, very large networks, and much more: **$795**

"BrainMaker Professional is a high-powered artificial neural network. I have tremendous respect for this program and its capabilities."
Stan Miastowski, BYTE

- **BrainMaker C25 Accelerator Board:** New price! Reduced more than 50% to only **$1995**
- **Intel i80170 and Micro Devices MD1220** neural network chip support. Build and train networks for these important new chips **now!**

30-day unconditional guarantee!
Free Catalog: (800) 284-8112
California Scientific Software
10141 Evening Star Dr #6, Grass Valley, CA 95945

CIRCLE 16 ON READER SERVICE CARD

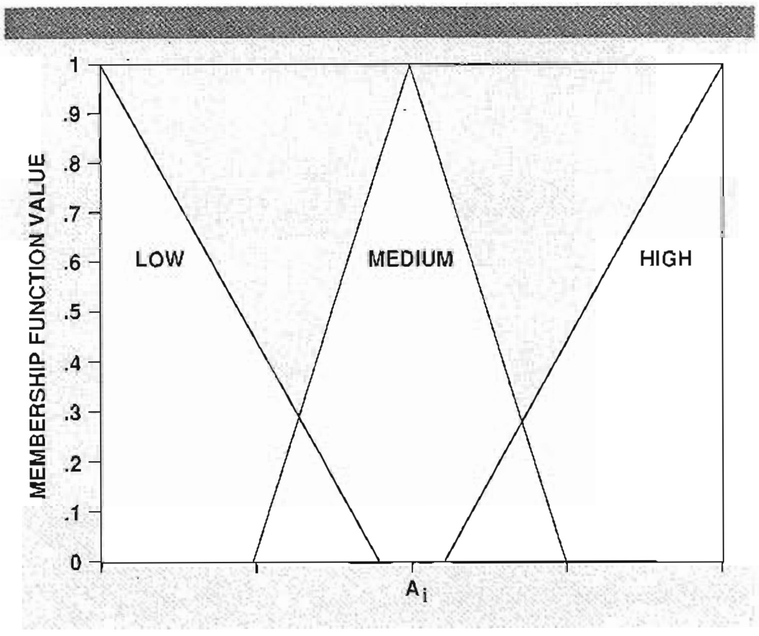


FIGURE 3. Triangular-shaped membership functions are easily altered using a GA.

that the best solution in a particular generation will always be at least as good as any solutions that have been encountered in previous ones.

The systematic information exchange utilizing probabilistic decisions is implemented by the second operator, crossover. Crossover provides a mechanism for strings to mix and match their desirable qualities through a random process. After reproduction, simple crossover proceeds in three steps. First, two newly reproduced strings are selected from the mating pool formed through reproduction. Second, a position along the two strings is selected uniformly at random. For example, the following binary coded strings *A* and *B* of length 10 are shown aligned for crossover:

```
A = 1 1 0  1 0 1 0 0 0 0
B = 0 0 1  0 1 1 1 1 1 1.
```

Notice how crossing site three has been selected in this particular example through random choice, although any of the other eight positions were just as likely to have been selected.

The third step is to exchange all characters following the crossing site. *A'* and *B'* are two new strings following this crossing:

```
A' = 1 1 0 0 1 1 1 1 1 1.
B' = 0 0 1 1 0 1 0 0 0 0.
```

String *A'* is made up of the first part of string *A* and the tail of string *B* Likewise, string *B'* is made up of the first part of string *B* and the tail of string *A*. Although crossover has a random element, it should not be thought of as a random walk through the search space. When combined with reproduction, it is an effective means of exchanging information and combining portions of high quality solutions.

Reproduction and crossover give GAs most of their search power. The third operator, mutation, enhances a GA's ability to find near optimal solutions. Mutation is the occasional alteration of a value at a particular string position, an insurance policy against the permanent loss of any simple bit. A generation may be created void of a particular character at a given string position. For example, a generation may exist that does not have a one in the third string position when, due to the chosen coding, a one in the third position may be critical to obtaining a quality solution.

Under these conditions, neither reproduction nor crossover will ever produce a one in this third position in subsequent generations. Mutation, however, causes a zero in the third position to be changed to a one occasionally. Thus, the critical piece of information can be reinstated into the population. Although mutation can serve a vital role in a GA, it occurs with a small probability (on the order of one mutation per 1,000 string positions) and is secondary to reproduction and crossover.

FLC VS. GA

The basic objective of the Bureau of Mines' research was to develop a robust technique for designing FLCs. This design process was executed either by establishing a set of rules and selecting membership functions, or by establishing a set of membership functions and selecting a set of rules. GAs demonstrate many characteristics that make them inviting for this task. When applying a GA to a search problem, two decisions have to be made: how to code the possible solutions to the problem as finite bit strings and how to evaluate the merit of each string.

Consider an approach to coding the FLC design problem in which the rule set has been established. In this problem, high-performance membership functions are sought. These membership functions can be represented in any of a number of ways. However, they are generally represented with functional relationships of one kind or another. For the purpose of this discussion, the degree of membership will be considered to be defined by a triangular-shaped membership function (Figure 3).

The only constraint placed on the individual triangles is that the triangles bordering the extreme limits of the action or control variables (extreme triangles like low and high) must remain right triangles, while the triangles not bordering the extreme limits of the action or control variables (interior triangles like medium) must remain isosceles triangles. Other than that, they can distort (variable base width) and translate (shift along the x-axis) freely. Therefore, each extreme triangle requires the definition of only one point to fix it. Each interior triangle requires the definition of two points to fix it (both base points).

To code the problem adequately, the entire set of membership functions necessary to drive an FLC must be represented as a bit string. This task is easily accomplished with a common coding called a concatenated, mapped, unsigned binary coding.

As for the second decision, the bit strings representing the parameters of the search problem must be judged and assigned a score (a fitness-function value) representing the degree to which they accomplish the goal of defining a high-performance FLC. For most systems, this goal requires driving the system to a setpoint and maintaining that setpoint until told to do otherwise.

Some squared-error term can be evaluated where the error is the distance between the setpoint and the state of the system. This aspect of the technique borders more on art than science. The development of a fitness function that entices the GA to locate membership functions that drive the physical system to the setpoint as rapidly as possible is application-dependent. **AI**

SUGGESTED READING

De Jong, K. A. "Analysis of the Behavior of a Class of Genetic Adaptive Systems." Ph.D. Thesis, Ann Arbor, Mich: University of Michigan, 1975. An early empirical investigation of the efficacy of GAs in function optimization. The set of functions has become a benchmark for GA operators.

Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, Mass: Addison-Wesley, 1989. Almost the principal reference on genetic algorithms, the book begins with the mechanics of GAs and discusses theory and applications, such as function minimization and machine learning.

Holland, J. H. *Adaptation in Natural and Artificial Systems.* Ann Arbor, Mich: University of Michigan Press, 1975. The seminal work on genetic algorithms, this book is highly technical, but a must for those interested in a detailed theoretical discussion.

Hollstien, R. B. "Artificial Genetic Adaptation in Computer Control Systems." Ph.D. Thesis, University of Michigan, 1971. Like De Jong's work, an early empirical investigation of GAs in function optimization.

Sugeno, M., Ed. *Industrial Applications of Fuzzy Control.* Amsterdam: Elsevier, 1985. Each paper in this collection describes the implementation of a FLC to a specific problem. Excellent for first-time developers of FLCs.

Zadeh, L. A. "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes." *IEEE Transactions on Systems, Man and Cybernetics* SMC-3(1), 1973: 28-44. An early work by the inventor of fuzzy logic. It may be too technical for those interested only in developing a FLC.

Chuck Karr is a mechanical engineer with the U.S. Bureau of Mines and works at the Tuscaloosa Research Center in Alabama. His research has focused on the adaptive control of mineral beneficiation processes using GAs and fuzzy logic. He is exploring the use of GAs in the design of mineral-process equipment.

Bart Kosko with Fred Watkins

on

"Fuzzy Systems and Applications"

A Three day seminar on an important technology for commercial, industrial and defense problems. If your problems are very complex, or cannot be represented by a series of IF-THEN rules, then fuzzy logic may be the way to go. Learn about fuzzy logic, how it is a methodology that simulates human thinking by incorporating the imprecision inherent in all physical systems, why it is very powerful in fields such as expert systems, decision-making, and information processing, and, how it can help you solve today's problems. Emphasis on many real problems and on a graphical development tool for fuzzy systems.

❖ Los Angeles, CA: June 5-7, 1991 & Washington, D.C. Area: June 17-19, 1991

☞ **For a detailed seminar brochure write to:** United Signals & Systems, Inc., P. O. Box 539, Needham, MA 02192; or, **call or FAX** (617) 444-7355

CIRCLE 17 ON READER SERVICE CARD