Maximizing Sets and Fuzzy Markoff Algorithms

Lotfi A. Zadeh, *Life Fellow, IEEE*

Abstract—A fuzzy algorithm is an ordered set of fuzzy instructions that upon execution yield an approximate solution to a given problem.

Two unrelated aspects of fuzzy algorithms are considered in this paper. The first is concerned with the problem of maximization of a reward function. It is argued that the conventional notion of a maximizing value for a function is not sufficiently informative and that a more useful notion is that of a maximizing set. Essentially, a maximizing set serves to provide information not only concerning the point or points at which a function is maximized, but also about the extent to which the values of the reward function approximate to its supremum at other points in its range.

The second is concerned with the formalization of the notion of a fuzzy algorithm. In this connection, the notion of a fuzzy Markoff algorithm is introduced and illustrated by an example. It is shown that the generation of strings by a fuzzy algorithm bears a resemblance to a birth-and-death process and that the execution of the algorithm terminates when no more "live" strings are left.

I. Preamble

THIS PAPER is based on a report entitled "On Fuzzy Algorithms," [14] or FA for short, which was written in 1972 but not submitted for publication at that time.

The report dealt with two issues that have grown in importance in the intervening years. The first issue relates to the concept of maximization (or minimization). The issue arises because in many real-world problems what matters is not only the value of x, say x_0, at which a function $f(x)$ is maximized, but also the robustness of the maximizing value. This suggests that the concept of a maximizing value be replaced by the concept of a maximizing fuzzy set or, more simply, a maximizing set. This is the issue that is addressed in the first part of FA.

The second part of FA deals with the concept of a fuzzy algorithm. In an informal way, the concept of a fuzzy algorithm was introduced in [13]. One of the main objectives of FA is to formalize the concept of a fuzzy algorithm, employing for this purpose the concept of a Markoff algorithm, rather than that of a Turing machine [1], [2], [11]. An interesting aspect of fuzzy Markoff algorithms relates to the fact that, as in the case of genetic algorithms [6], [9], [10], the result of execution at each stage is a set of strings, rather than a single string. However, in the case of fuzzy Markoff algorithms, the set of strings is fuzzy, rather than crisp. Fuzzification of the

concept of a genetic algorithm would yield the same result. A natural way of fuzzifying—and hence generalizing—the concept of a genetic algorithm is to assume that the fitness function is both fuzzy and granular, rather than crisp. Such an assumption would be closer to reality.

The importance of the concept of a fuzzy algorithm stems from the fact that fuzzy algorithms mimic the ways in which humans make decisions and act in the presence of imprecision, uncertainty, and partial truth. But what is even more important is that fuzzy algorithms provide a more realistic way of defining many basic concepts that are intrinsically fuzzy, e.g., the concepts of smoothness, ovalness, stationarity, independence, stability, and causality. This is the issue that was addressed in subsequent papers and especially in [15] and [16], in which the basic concepts of a linguistic variable, fuzzy if–then rule, and fuzzy questionnaire were introduced. In particular, some of the ideas introduced in [16] relate to later theories on rough sets [7] and machine learning [8].

II. Introduction

Roughly speaking, a fuzzy algorithm [13] is an ordered set of fuzzy instructions that upon execution yield an approximate solution to a given problem. As in the case of nonfuzzy algorithms, a fuzzy algorithm is usually expected to be capable of providing an approximate solution to any problem in a specified class of problems, rather than to a single problem.

Simple examples of fuzzy algorithms that occur in everyday experience are cooking recipes, instructions for parking a car, instructions for tying a knot, etc. As a more concrete example, consider the following simple control problem. Suppose that we wish to transfer a blindfolded subject A from an initial position x in a room with no obstacles to a final position y. Furthermore, suppose that the fuzzy instructions are limited to the following set: 1) turn counterclockwise by approximately α degrees, with α being a multiple of, say, 15; 2) take a step; 3) take a small step; and 4) take a very small step.

Under these assumptions, a simple fuzzy algorithm for guiding A from x to y may be stated as follows. (It is understood that after A executes an instruction, his or her new position and orientation are observed fuzzily by the experimenter, who then chooses that instruction from the algorithm that most closely fits the last observation). Basically, the algorithm mimics what a human might do to reach the target.

1) If A is facing y, then go to 2, else go to 5.
2) If A is close to y, go to 7, else go to 6.
3) If A is very close to y, go to 8, else go to 7.
4) If A is very very close to y, then stop.
5) Ask A to turn counterclockwise by an amount needed to make him or her face y. Go to 1.

6) Ask A to take a step. Go to 1.

7) Ask A to take a small step. Go to 1.

8) Ask A to take a very small step. Go to 1.

Will this algorithm work? If yes, how well? A basic characteristic of fuzzy algorithms is that questions of this type cannot, in general, be answered precisely. Thus, we must be content with fuzzy answers, such as "the algorithm will work reasonably well so long as the degree of fuzziness in observations is relatively small and the subject executes the instructions in a way that is consistent with the expectation of the experimenter." Needless to say, such vague assertions would not be acceptable to those who expect the convergence properties of an algorithm to be expressible as a provable theorem. Unfortunately, unpalatable as it may be, there may be no alternative to accepting much lower standards of precision if we wish to be able to devise approximate (that is, fuzzy) solutions to the many complex and ill-defined problems that arise in the analysis of large-scale man-machine and manlike systems [3]. Fuzzy algorithms and fuzzy algorithmic definitions may well prove to be of considerable practical importance once we learn more about their properties and how to construct them for specific purposes. In what follows, we shall focus our attention on a few concepts that may contribute to this objective.

III. THE CONCEPT OF A MAXIMIZING SET

Consider a real-valued function f on $X = \{x\}$, with $f(x)$ representing the reward associated with an action x, $x \in X$. We assume that f is bounded both from above and from below, that is, $-\infty < \operatorname{Inf} f \leq \operatorname{Sup} f < \infty$, where $\operatorname{Sup} f$ and $\operatorname{Inf} f$ represent, respectively, the supremum and infimum of f over X.

Suppose that we wish to maximize the reward and pose the question: for what value of x does f attain its maximum value? A conventional answer to this question might be "at $x = x_0$." It is clear, however, that such an answer does not provide sufficient information about the value that should be assigned to x because what matters is not only that f is maximized at $x = x_0$, but also how it behaves in the neighborhood of x_0. Thus, if f is quite flat around x_0, then the solution $x = x_0$ is a robust one, and hence, it is not essential that x be exactly equal to x_0. The opposite is true, of course, if f is sharply peaked around x_0.

The inadequacy of the concept of a maximizing value suggests the introduction of a more general concept, namely, the concept of a *maximizing set*. Intuitively, a maximizing set $M(f)$, or simply M, for a function f on X is a fuzzy subset of X such that the grade of membership of a point x in M represents the degree to which $f(x)$ approximates to $\operatorname{Sup} f$ in some specified sense. For example, suppose for simplicity that X is the finite set $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and that $f(1) = 5$, $f(2) = 6$, $f(3) = 8$, $f(4) = 10$, $f(5) = 9$, $f(6) = 2$, $f(7) = 6$, and $f(8) = 1$. Further, suppose that the grade of membership of x in M is defined by

$$\mu_M(x) = \left[\frac{f(x)}{\operatorname{Sup} f}\right]^2, \quad \text{if } \mu_M x > 0.5$$
$$= 0, \qquad\qquad \text{if } \mu_M x \leq 0.5. \qquad (1)$$

Then, $\mu_M(3) = 0.64$, $\mu_M(4) = 1$, and $\mu_M(5) = 0.81$, with all other points having zero grade of membership in M.

In the case under consideration, the maximizing value of x is $x_0 = 4$ and the maximizing set for f is the fuzzy set $M = \{(3, 0.64), (4, 1.0), (5, 0.81)\}$. Clearly, since the maximizing set provides essential information about the effect of choosing values of x other than x_0 on the values of the reward function f, it would be very desirable, in general, to know M—and not just x_0—in situations involving a decision on the value to assign to x in order to maximize a reward function. For example, if the maximizing set M for a reward function defined on the interval $[0, 10]$ is of the form

$$\mu_M(x) = 0.9, \quad \text{for } 0 \leq x \leq 5$$
$$\mu_M(x) = 0.2, \quad \text{for } 5 < x < 6$$
$$\mu_M(x) = 1.0, \quad \text{for } x = 6$$
$$\mu_M(x) = 0.1, \quad \text{for } 6 < x \leq 10$$

then the maximizing value for the reward is $x_0 = 6$. However, inasmuch as the values of the reward function in the immediate neighborhood of $x_0 = 6$ are very low, it would be very risky to set $x = 6$. Obviously, it would be much better to set $x = 3$, say, since μ_M is flat and close to unity in the neighborhood of three. Although contrived to illustrate the point, the example clearly shows the inadequacy of the notion of the maximizing value for dealing with realistic decision problems—problems in which the sensitivity of the solution to perturbations is almost always an important issue.

An equation such as (1) serves to "calibrate" the definition of a maximizing set. It would be unreasonable to expect that a universal definition be applicable to all situations. However, the following calibrating definition can frequently be used as a starting point from which other definitions, if necessary, may be obtained by modification.

A. Definition of a Maximizing Set

Assume that a real-valued reward function f is defined over a domain X and that $-\infty < \operatorname{Inf} f \leq \operatorname{Sup} f < \infty$, that is, f is bounded both from above and from below. To define a maximizing set for f, it is expedient to consider separately three cases, as follows.

Case 1: A reward function f will be said to be positive-definite if it is nonnegative for all x in X, that is, if $\operatorname{Inf} f \geq 0$ (Fig. 1). In this case, the membership function of the maximizing set is defined by

$$\mu_M(x) = \frac{f(x)}{\operatorname{Sup} f}, \quad x \in X. \qquad (2)$$

Case 2: A reward function f is nondefinite if $\operatorname{Inf} f \leq 0$ and $\operatorname{Sup} f \geq 0$ (Fig. 2). In this case, the defining relation (2) is applied to a translate of f, expressed by

$$f^* = f - \operatorname{Inf} f. \qquad (3)$$

Thus, for a nondefinite reward function, the membership function of the maximizing set is defined by

$$\mu_M(x) = \frac{f - \operatorname{Inf} f}{\operatorname{Sup} f - \operatorname{Inf} f}. \qquad (4)$$
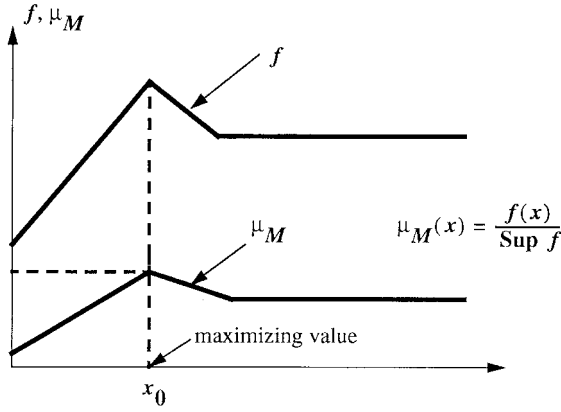
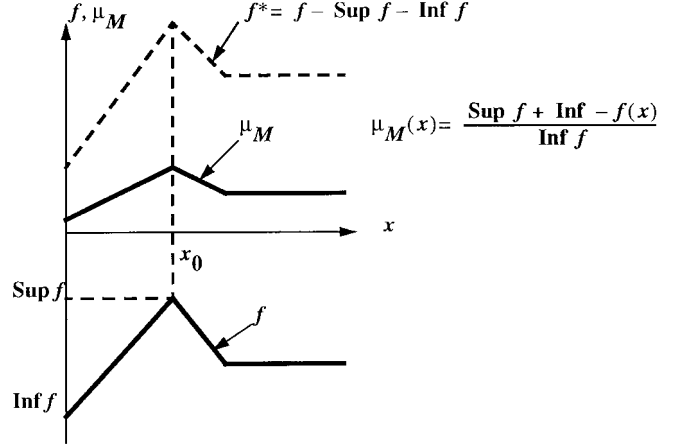Fig. 1. Maximizing set for a positive-definite reward function.

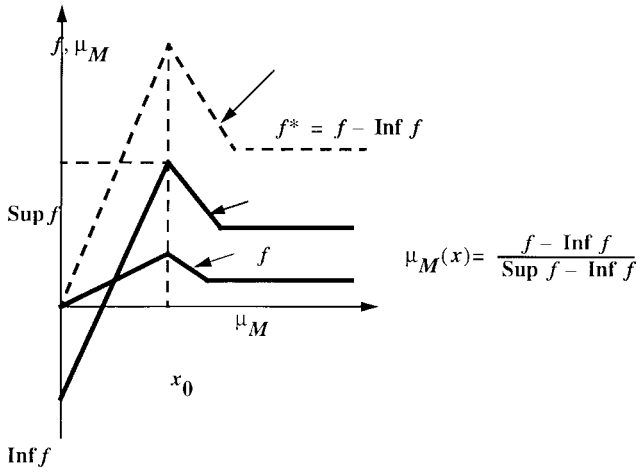

Fig. 2. Maximizing set for a nondefinite reward function.

Fig. 3. Maximizing set for a negative-definite reward function.

Case 3: A reward function f is negative-definite if it is nonpositive for all x in X, that is, $\operatorname{Sup} f \leq 0$. In this case, (2) is applied to a translate of f, expressed by

$$f^* = f - \operatorname{Sup} f - \operatorname{Inf} f \qquad (5)$$

yielding the definition

$$\mu_M(x) = \frac{\operatorname{Sup} f + \operatorname{Inf} f - f}{\operatorname{Inf} f}. \qquad (6)$$

Taken together, (2), (4), and (6) constitute a general definition of the maximizing set for a reward function that is bounded both from above and from below.

The following properties of the maximizing set are immediate consequences of the above definition.

1) The maximizing set for f is unique.
2) The maximizing set for f is invariant under linear scaling.

   In other words

$$M(kf) = M(f) \qquad (7)$$

   where k is any real constant.

3) The maximizing set for f is not invariant under translation.

Thus, if $f \neq 0$ and $\alpha = \text{constant} \neq 0$, then in general

$$M(f + \alpha) \neq M(f).$$

More specifically, for any fixed x, the dependence of $\mu_M(x)$ on α is of the form

$$\mu_M(x) = \frac{a + \alpha}{b + \alpha}$$

where a and b are constants. This implies that, as the magnitude of α increases, the membership function tends to unity.

4) For $X = R^n$, the maximizing set is convex if and only if the reward function is quasiconcave.

   By definition, a fuzzy subset A of R^n (linear vector space of n-tuples of real numbers) is convex [12] if all of its level sets are convex, that is, if the sets

$$A_\alpha = \{x \mid \mu_A(x) \geq \alpha\} \qquad (8)$$

   are convex in R^n. Equivalently, A is convex if μ_A satisfies

$$\mu_A(\lambda x_1 + (1 - \lambda)x_2) \geq \operatorname{Min}(\mu_A(x_1), \mu_A(x_2)) \qquad (9)$$

   for all λ in $[0, 1]$ and all x_1, x_2 in R^n.

   Also by definition, a function f on R^n is quasiconcave if all of the level sets

$$f_\alpha = \{x \mid f(x) \geq \alpha\}$$

   are convex for all finite α in $(-\infty, \infty)$.

Now the definition of μ_M in terms of f implies that if f is quasiconcave, so is μ_M. This in turn implies the convexity of M. Thus, if f is quasiconcave, then M is convex and vice-versa.

Example: The maximizing sets for the reward functions shown in Figs. 1–3 are convex.

Comment: It should be observed that a concave reward function can be transformed into a quasiconcave function that is bounded both from above and from below by an order-preserving transformation.

B. The Fuzzy Maximum

The concept of a maximizing set is, in effect, a fuzzification of the concept of a maximizing value. From this point of view, it is natural to ask the question: what is the fuzzy counterpart of the notion of the maximum of a function? To put it another way: if x_0 is a maximizing value for f, then the maximum value of f is $f(x_0)$, that is, the image of x_0 under the mapping f. What is the corresponding image of the maximizing set M?

To answer this question, we note that if f is a function from $X = \{x\}$ to $Y = \{y\}$, with $y = f(x)$, then, as defined in [12], a fuzzy set A in X induces a fuzzy set B in Y, whose membership function is given by

$$\mu_B(y) = \operatorname*{Sup}_{x \in f^{-1}(y)} \mu_A(x) \tag{10}$$

where $f^{-1}(y)$ is the preimage of y, that is

$$f^{-1}(y) = \{x \mid y = f(x)\}. \tag{11}$$

If we identify A with the maximizing set in X, then B may be interpreted as the fuzzy maximum of f, which is a fuzzy subset of Y ($Y =$ range of f). Denoting this fuzzy subset of Y by MM, it follows from the definition of M that the value of $\mu_M(x)$ for each point in the preimage of f is the same, namely, $\mu_M(f^{-1}(y))$. Consequently, from (10), we can infer that the membership function of the fuzzy maximum of f is given by

$$\mu_{MM}(y) = \mu_M(f^{-1}(y)), \quad \text{if } f^{-1}(y) \text{ exists}$$
$$= 0, \quad \text{otherwise.} \tag{12}$$

Example: Suppose that $X = [0, 10]$ and f has the form shown in Fig. 1. More specifically

$$f(x) = x + 2, \quad 0 < x < 2$$
$$f(x) = -x + 6, \quad 2 < x < 3$$
$$f(x) = 3, \quad x > 3.$$

The maximizing set for f is given by

$$\mu_M(x) = \frac{x+2}{4}, \quad 0 \leq x \leq 2$$
$$\mu_M(x) = \frac{6-x}{4}, \quad 2 \leq x \leq 3$$
$$\mu_M(x) = \frac{3}{4}, \quad x > 3.$$

Correspondingly, the fuzzy maximum of f is given by

$$\mu_{MM}(y) = \frac{y}{4}, \quad 2 \leq y \leq 4$$
$$= 0, \quad \text{elsewhere.}$$

C. The Minimizing Set

So far we have focused our attention on the concept of the maximizing set for f, denoted by $M(f)$ or simply M. In terms of M, the minimizing set for f, denoted by $m(f)$ or simply m, may be defined by

$$\text{Minimizing set for } f = \text{Maximizing set for } -f. \tag{13}$$

By using the expressions for μ_M given by (2), (4), and (6), it can readily be shown that the sum of μ_M and μ_m is a constant. Thus

$$\mu_M(x) + \mu_m(x) = \frac{\operatorname{Sup} f + \operatorname{Inf} f}{\operatorname{Sup} f} \quad \text{for positive-definite } f \tag{14}$$

$$\mu_M(x) + \mu_m(x) = 1 \quad \text{for nondefinite } f \tag{15}$$

and

$$\mu_M(x) + \mu_m(x) = \frac{\operatorname{Sup} f + \operatorname{Inf} f}{\operatorname{Inf} f} \quad \text{for negative-definite } f. \tag{16}$$

The constancy of the sum of $\mu_M(x)$ and $\mu_m(x)$ implies that $\mu_M(x)$ is large where $\mu_m(x)$ is small and vice-versa. (Fuzzy sets that are related to one another in this way are weakly complementary.) In particular, if f is nondefinite, then by (15), M and m are complementary fuzzy subsets of X—a property that is in accord with our intuition.

The maximizing and minimizing sets of a reward function contain the type of information that is usually provided by sensitivity analysis. In practice, these sets would usually be defined by exemplification, that is, by associating approximate grades of membership with a finite set of representative points in X.

IV. FUZZY MARKOFF ALGORITHMS

In the preceding section, we were concerned with the formalization of the notion of an approximate maximizing value, which led us to the concept of a maximizing set.

In this section, our concern is with the formalization of the notion of a fuzzy algorithm. As was pointed out in [13], a fuzzy algorithm may be equated with a fuzzy Turing machine. In [11], both the fuzzy Turing machine and the fuzzy Markoff algorithm are defined and their equivalence is demonstrated. Here, we shall give a simpler definition of a fuzzy Markoff algorithm, which is a natural extension of the conventional way in which a Markoff algorithm is defined [1], [2]. In effect, our definition of a fuzzy Markoff algorithm is intended mainly to make more precise the concept of a fuzzy algorithm in the same sense that a Markoff algorithm formalizes the concept of a nonfuzzy algorithm.

Let A^* denote the set of all finite strings over a finite alphabet A. For our purposes, it will be convenient to represent a finite fuzzy subset L of A^* in the form of a linear combination

$$L = \mu_1 \alpha_1 + \mu_2 \alpha_2 + \cdots + \mu_n \alpha_n \tag{17}$$

where the α_i, $1, \ldots, n$, denote strings in A^* and the μ_i represent their respective grades of membership in L.

Example: Let $A = \{a, b\}$. Then

$$L = 0.3aab + 0.8bba + 1.0aa + 0.3ba$$

is equivalent to expressing L as the collection of ordered pairs

$$L = \{(0.3, aa(b), (0.8, bba), (1.0, aa), (0.3, ba)\}.$$

When expressed in the form (17), the concatenation of two fuzzy sets of strings

$$L = \mu_1\alpha_1 + \cdots + \mu_n\alpha_n$$

and

$$L = \mu'_1\alpha'_1 + \cdots + \mu'_m\alpha'_m$$

where $\alpha_1, \ldots, \alpha_n$ and $\alpha'_1, \ldots, \alpha'_m$ are strings in A^* that can readily be obtained by term-by-term multiplication and addition, with the understanding that

$$\mu_i\mu'_j = \mu_i \wedge \mu'_j = \text{Min}(\mu_1, \mu'_j) \tag{18}$$

$$\mu_j + \mu'_j = \mu_i \vee \mu'_j = \text{Max}(\mu_1, \mu'_j). \tag{19}$$

Example:

$$L = 0.3aa + 0.5aba$$
$$L' = 0.2ab + 0.9ba.$$

Then

$$LL' = 0.2aaab + 0.2abaab + 0.3aaba + 0.5ababa.$$

A nonfuzzy Markoff algorithm M is a function from A^* to A^* that is characterized by a finite sequence of productions $P_1, \ldots, P_n$ of the general form

$$\alpha \to \beta \quad \text{or} \quad \alpha \to \beta. \tag{20}$$

where α (the antecedent) and β (the consequent) are strings in A^*, and the arrow signifies that if α occurs as a substring in a string x, then the leftmost occurrence of α in x may be replaced by β. The presence of the period indicates that the production is terminal, in the sense that the execution of the algorithm terminates after a terminal production is applied.

A typical very simple problem in the theory of Markoff algorithms is the following. Suppose $A = \{a, b, c\}$, and let x be any string in A^*. Find a Markoff algorithm that removes the first three occurrences of c from x. For example, $acabbcabcc$ is transformed into $aabbabc$. A fuzzy version of this problem would be: find a fuzzy Markoff algorithm that removes the first few occurrences of c from x. In this case, if we define few as the fuzzy set

$$\text{few} = \{(1, 1.0), (2, 1.0)(3, 0.8)(4, 0.6)\}$$

then the result of applying this fuzzy Markoff algorithm to $x = acabbabcc$ would be a fuzzy set rather than a single string. Thus, denoting the fuzzy algorithm by FM, we have in symbols

$$\text{FM}(acabbcabcc)$$
$$= 1.0aabbcabcc + 1.0aabbabcc + 0.8aabbabc + 0.6aabbab.$$

More generally, let $\Im(A^*)$ denote the set of all fuzzy subsets of A^*. Then an FM may be regarded as a function from A^* to $\Im(A^*)$, which satisfies certain conditions and is characterized in a particular way that will be described presently.

Specifically, let L be a finite fuzzy subset of A^*

$$L = \mu_1\alpha_1 + \cdots \mu_n\alpha_n, \quad \alpha_i \in A^*, \quad i = 1, \ldots, n. \tag{21}$$

Then we postulate that the image of L under FM is given by

$$\begin{aligned} \text{FM}(L) &= \text{FM}(\mu_1\alpha_1 + \cdots + \mu_n\alpha_n) \\ &= \mu_1\text{FM}(\alpha_1) + \cdots + \mu_n\text{FM}(\alpha_n) \end{aligned} \tag{22}$$

which implies that FM is a linear operator in A^*. In consequence of (22), the operation of FM on a fuzzy set of strings can be described in terms of its operation on individual strings. This basic property of fuzzy Markoff algorithms plays an important role in the description of their execution. In contrast to the form of a production in a nonfuzzy Markoff algorithm, a typical production P_i in a fuzzy Markoff algorithm has the appearance

$$P_i : \alpha \to \mu_1\beta_1 + \cdots + \mu_k\beta_k + \mu_{k+1}\beta_{k+1} \cdot + \cdots + \mu_n\beta_n \tag{23}$$

where the $\beta's \in A^*$, the $\mu's$ are numbers in the interval $[0, 1]$, and the terms ending with a period are the terminating components in the consequent of P_i. The important point is that the consequent of p_i is a fuzzy set of strings rather than a single string.

Now suppose that a string x can be expressed as $x = u\alpha\nu$, where $u, \alpha, \nu \in A^*$ and α is not a substring of u. (i.e., α in $u\alpha\nu$ represents its leftmost occurrence.) Then, on substituting the consequent of p_i for α in x, we obtain the fuzzy set of strings represented by.

$$\begin{aligned} u\alpha\nu &= u(\mu_1\beta_1 + \cdots + \mu_k\beta_k + \mu_{k+1}\beta_{k+1} \cdot + \cdots + \mu_n u\beta_n.)\nu \\ &= \mu_1 u\beta_1\nu + \cdots + \mu_k u\beta_k\nu + \mu_{k+1}u\beta_{k+1}\nu. \\ &\quad \cdot + \cdots + \mu_n u\beta_n\nu. \end{aligned} \tag{24}$$

Furthermore, if ρ is a number in $[0, 1]$, then

$$\begin{aligned} \rho u\alpha\nu &= \rho\mu_1 u\beta_1\nu + \cdots + \rho\mu_k u\beta_k\nu \\ &\quad + \rho\mu_{k+1}u\beta_{k+1}\nu \cdot + \cdots + \rho\mu_n u\beta_n\nu \end{aligned} \tag{25}$$

where

$$\rho\mu_i = \rho \wedge \mu_i, \quad i = 1, \ldots, n.$$

Example: Suppose $x = abbabbb$ and

$$ba \to 0.3ab + 0.9a.$$

Then

$$abbabbb \neq 0.3ababbbb + 0.9ababbb.$$

and

$$0.4abbabbb \neq 0.3ababbbb + 0.4ababbb.$$

In summary, a production P_i is applicable to a string x if its antecedent is a substring of x. The result of applying P_i to x is expressed by (24) and (25). (Note that the substitution is made in the leftmost occurrence in x of the antecedent of P_i.)

We are now ready to define the execution of a fuzzy Markoff algorithm in terms of (22) and the rewriting rules (24) and (25).

Definition of a Fuzzy Markoff Algorithm: A fuzzy Markoff algorithm is a function from A^* to $\Im(A^*)$ that satisfies (22) and is characterized by (a) a finite alphabet A, (b) possibly a finite auxiliary alphabet A' (comprising various markers that may be needed for bookkeeping purposes), and (c) a finite sequence of productions $P_1, \ldots, P_n$ of the form (23), with $P_n : \Lambda \to \Lambda$. ($\Lambda$ = null string).

It is convenient to describe the operation of FM on a string x in A^* in terms of a subalgorithm $\overline{\text{FM}}$, which is defined on $\Im(A^*)$. Thus, if L is a fuzzy set of strings in A^*, e.g.,

$$L = \mu_1 \nu_1 + \cdots + \mu_r \nu_r, \quad \nu_i \in A^*, \quad i = 1, \ldots, r$$

then, as in (22)

$$\overline{\text{FM}}(L) = \mu_1 \overline{\text{FM}}(\nu_1) + \cdots + \mu_r \overline{\text{FM}}(\nu_r). \tag{26}$$

To compute $\overline{\text{FM}}(\nu_i), \nu_i \in A^*$, $i = 1, \ldots, r$, we proceed as follows.

Apply the first applicable production in the sequence $P_1, \ldots, P_n$, to ν_i and call the result $\text{FM}(\nu_i)$. (Note that P_n will always be applicable since ν_i may be written as $\Lambda \nu_i$.) Set

$$D(\nu_i) = \text{sum of terms in } \overline{\text{FM}}(\nu_i) \text{ that terminate in a}$$
$$\text{period} \tag{27}$$
$$B(\nu_i) = \text{sum of the remaining terms in } \overline{\text{FM}}(\nu_i) \tag{28}$$

and

$$D(L) = \mu_1 D(\nu_1) + \cdots + \mu_r D(\nu_r) \tag{29}$$
$$B(L) = \mu_1 B(\nu_1) + \cdots + \mu_r B(\nu_r). \tag{30}$$

The result of operating with $\overline{\text{FM}}$ on L is defined to be the sum of the fuzzy sets of strings $D(L)$ and $B(L)$. Thus

$$\overline{\text{FM}}(L) = D(L) + B(L). \tag{31}$$

In terms of $\overline{\text{FM}}$, the computation of $\text{FM}(x)$, where $x \in A^*$, is carried out as follows.

1) Set $L = x = $ initial string.
2) Set $\text{FM}(x) = \theta$ (empty set).
3) Apply $\overline{\text{FM}}$ to L and compute $\overline{\text{FM}}(L) = B(L) + D(L)$.
4) Set

$$\text{FM}(x) = \text{FM}(x) + D(L). \tag{32}$$

5) If $B(L)$ is empty, terminate the execution of FM.
6) If $B(L)$ is not empty, set $L = B(L)$. Go to 3.

The execution of the algorithm is actually quite straightforward and simple in principle. The following example will serve as an illustration. (No auxiliary alphabet used.)

Example: Assume $A = \{a, b\}$ and

$$P_1 : \quad ab \to 0.3bb + 0.6a + 0.9ab.$$
$$P_2 : \quad ba \to 0.8a + 1.0bb.$$
$$P_3 : \quad\;\; b \to 0.6a.$$
$$\Lambda \to 1.0\Lambda.$$

Let the initial string be $x = abba$. Applying $\overline{\text{FM}}$ to $L = x$, we note that the first applicable production is P_1. It yields for

$L = x$

$$\overline{\text{FM}}(L) = 0.3bbba + 0.6aba + 0.9abba.$$
$$D(L) = 0.9abba.$$
$$B(L) = 0.3bbba + 0.6aba$$

and

$$\text{FM}(x) = 0.9abba.$$

Since $B(L)$ is nonempty, we apply $\overline{\text{FM}}$ to $L = B(L)$, yielding

$$\overline{\text{FM}}(0.3bbba + 0.6aba) = 0.3\overline{\text{FM}}(bbba) + 0.6\overline{\text{FM}}(aba).$$

Now, applying P_2 to $bbba$, we get

$$\overline{\text{FM}}(bbba) = 0.8bbba. + 1.0bbbb.$$

and applying P_1 to aba, we have

$$\overline{\text{FM}}(aba) = 0.3bba + 0.6aa + 0.6aba.$$

Thus

$$\text{FM}(L) = 0.3bba + 0.6aa + 0.6aba. + 0.3bba. + 0.3bbbb.$$
$$D(L) = 0.6aba. + 0.3bba. + 0.3bbbb.$$
$$B(L) = 0.3bba + 0.6aa$$

and

$$\text{FM}(x) = 0.9abba. + 0.6aba. + 0.3bba. + 0.3bbbb.$$

Since $B(L)$ is nonempty, we apply $\overline{\text{FM}}$ to $L = B(L)$, obtaining

$$\overline{\text{FM}}(0.3bba + 0.6aa) = 0.3\overline{\text{FM}}(bba) + 0.6\overline{\text{FM}}(aa).$$

Next, applying P_2 to bba, we get

$$\overline{\text{FM}}(bba) = 0.8ba. + 1.0bbb.$$

and applying P_3 to aa, we obtain

$$\overline{\text{FM}}(aa) = 1.0aa.$$

Thus

$$\overline{\text{FM}}(L) = 0.3ba. + 0.3bbb. + 0.6aa.$$
$$D(L) = 0.3ba. + 0.3bbb. + 0.6aa.$$
$$B(L) = \theta$$

and finally

$$\overline{\text{FM}}(x) = 0.9abba + 0.6aba + 0.3bba.$$
$$+ 0.3bbbb. + 0.3ba. + 0.3bbb. + 0.6aa.$$

At this point, execution of the algorithm terminates because $B(L) = \theta$.

The execution of an FM may be likened to a birth-and-death process in which the operation with $\overline{\text{FM}}$ on a string ν_i gives rise to the birth of new strings, represented by $B(\nu_i)$, and the death of others, represented by $D(\nu_i)$. In the same sense, $B(L)$ and $D(L)$ represent the "newly born" and the "dead" strings resulting from operating with $\overline{\text{FM}}$ on a "live" fuzzy set of strings L. Finally, $\text{FM}(x)$ plays the role of the population of

the dead in a cemetery with (32) representing the addition of the newly deceased to that population.

As in a birth-and-death process, the population of "live" strings can grow explosively if the productions $P_1, \ldots, P_n$ are such that each execution of $\overline{FM}$ results in significantly more "births" than "deaths." This rather interesting aspect of fuzzy Markoff algorithms is not present in conventional Markoff algorithms.

In the foregoing discussion, we have restricted ourselves to formulating what appears to be a natural extension of the notion of a Markoff algorithm. Exploration of the properties of such algorithms will be pursued in subsequent papers on this subject.

REFERENCES

[1] A. T. Berztiss, *Data Structures*. New York: McGraw-Hill, 1971.
[2] S. S. L. Chang and L. A. Zadeh, "Fuzzy mapping and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-2, pp. 30–34, 1972.
[3] Z. Chi, H. Yan, and T. Pham, *Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition*. Singapore: World Scientific, 1996.
[4] H. Galler and A. Perlis, *A View of Programming Languages*. Reading, MA: Addison-Wesley, 1970.
[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[6] C. Z. Janikow, "A genetic algorithm method for optimizing fuzzy decision trees," *Inf. Sci.*, vol. 90, pp. 1–22, 1995.
[7] R. Pawlak, *Rough Sets—Theoretical Aspects of Reasoning About Data*. Boston, MA: Kluwer, 1990.
[8] J. R. Quinlan, *Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
[9] E. Sanchez, T. Shibata, and L. A. Zadeh, *Genetic Algorithms and Fuzzy Logic Systems Soft Computing Perspectives*. Singapore: World Scientific, 1997.
[10] S. K. Pal and P. P. Wang, *Genetic Algorithms for Pattern Recognition*. Boca Raton, FL: CRC, 1996.
[11] E. Santos, "Fuzzy algorithms," *Inf. Control*, vol. 17, pp. 326–339, 1970.
[12] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, pp. 338–353, 1965.
[13] _____, "Fuzzy algorithms," *Inf. Control*, vol. 12. pp. 99–102, 1968.
[14] _____, "On fuzzy algorithms," , Elec. Res. Lab., Univ. California, Berkeley, Memo. ERL-M325, 1972.
[15] _____, "Outline of a new approach to the analysis of complex systems and decision process," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 28–44, 1973.
[16] _____, "A fuzzy-algorithmic approach to the definition of complex or imprecise concepts," *Int. J. Man-Mach. Stud.*, vol. 8, pp. 249–291, 1976.
[17] _____, "Fuzzy logic, neural networks and soft computing," *Commun. ACM*, vol. 37, no. 3, pp. 77–84, 1994.
[18] _____, "Soft computing and fuzzy logic," *IEEE Software*, vol. 11, nos. 1–6, pp. 48–56, 1994.
[19] _____, "The roles of soft computing and fuzzy logic in the conception, design and deployment of intelligent systems," *Br. Telecom J.*, vol. 14, no. 4, 1996.
[20] _____, "Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic," *Fuzzy Sets Syst.*, vol. 90, pp. 111–128, 1997.

Lotfi A. Zadeh (S'45–A'47–M'47–SM'56–F'58–LF'87), for a photograph and biography, see this issue, p. 8.