ON FUZZY ALGORITHMS

by

L. A. Zadeh

Memorandum No. ERL-M325

22 February 1972

ON FUZZY ALGORITHMS

by

L. A. Zadeh

Memorandum No. ERL-M325

22 February 1972

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

ON FUZZY ALGORITHMS

L. A. Zadeh*

ABSTRACT

A fuzzy algorithm is an ordered set of fuzzy instructions which 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 Markoff 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.

---

*Department of Electrical Engineering and Computer Sciences, and the Electronics Research Laboratory, University of California, Berkeley, California 94720.

This work was supported in part by the Navy Electronic Systems Command, Contract N00039-71-C-0255.

I. Introduction

Roughly speaking, a fuzzy algorithm [1] is an ordered set of fuzzy instructions which upon execution yield an approximate solution to a given problem. As in the case of non-fuzzy 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 which 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 blind-folded 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 counter-clockwise 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 new position and orientation are observed fuzzily by the experimenter, who then chooses that instruction from the algorithm which fits most closely the last observation.)

1. If A is facing y then go to 2 else go to 5.
2. If A is close to y go to 3 else go to 7.
3. If A is very close to y go to 4 else go to 8.
4. If A is very very close to y then stop.

5. Ask A to turn counterclockwise by an amount needed to make him 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, one 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 which arise in the analysis of large-scale man-machine and man-like systems [2].

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 which may contribute to this objective.

2. 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, $-\infty < \text{Inf } f \leq \text{Sup } f < \infty$, where Sup f and 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 $x = x_0$," say. 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 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

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

$$= 0 \qquad \text{if} \quad \mu_M(x) \leq 0.5$$

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

In the case under consideration, the maximizing value of x is $x_o = 4$ 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_o on the values of the reward function f, it would be very desirable, in general, to know M - and not just x_o - 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

$$\mu_M(x) = 0.9 \qquad \text{for } 0 \leq x \leq 5$$

$$\mu_M(x) = 0.2 \qquad \text{for } 5 < x < 6$$

$$\mu_M(x) = 1.0 \qquad \text{for } x = 6$$

$$\mu_M(x) = 0.1 \qquad \text{for } 6 < x \leq 10$$

then the maximizing value for the reward is $x_o = 6$. However, inasmuch as the values of the reward function in the immediate neighborhood of $x_o = 6$ 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 3. 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.

Definition of a maximizing set

Assume that a real-valued reward function f is defined over a domain X and that $-\infty < \text{Inf } f \leq \text{Sup } f < \infty$, 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 non-negative for all x in X, that is, if $\text{Inf } f \geq 0$ (Fig. 1). In this case the membership function of the maximizing set is defined by

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

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

$$f^* = f - \text{Inf } f \tag{3}$$

Thus, for a non-definite reward function, the membership function of the

maximizing set is defined by

$$\mu_M(x) = \frac{f - \text{Inf } f}{\text{Sup } f - \text{Inf } f} \tag{4}$$

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

$$f^* = f - \text{Sup } f - \text{Inf } f \tag{5}$$

yielding the definition

$$\mu_M(x) = \frac{\text{Sup } f + \text{Inf } f - f}{\text{Inf } f} \tag{6}$$

Taken together, (2), (4) and (6) constitute a general definition of the maximizing set for a reward function which 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

$$M(kf) = M(f) \tag{7}$$

where k is any real constant.

3. The maximizing set for f is not invariant under translation. Thus, if $f \not\equiv 0$ and α = 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 quasi-concave.

By definition, a fuzzy subset A of R^n (linear vector space of n-tuples of real numbers) is convex [3] iff all of its level sets are convex, that is, iff the sets

$$A_\alpha = \{x | \mu_A(x) \geq \alpha\}, \qquad 0 < \alpha \leq 1 \tag{8}$$

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

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

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 quasi-concave iff all of the level sets

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

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

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

Example. The maximizing sets for the reward functions shown in Figs. 1, 2 and 3 are convex.

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

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_o is a maximizing value for f then the maximum value of f is $f(x_o)$, that is, the image of x_o 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 [3], a fuzzy set A in X induces a fuzzy set B in Y whose membership function is given by

$$\mu_B(y) = \sup_{x \in f^{-1}(y)} \mu_A(x) \tag{10}$$

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

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

$$= 0 \quad \text{otherwise.}$$

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

$$f(x) = x + 2 \quad , \quad 0 \leq x \leq 2$$

$$f(x) = -x + 6 \quad , \quad 2 < x \leq 3$$

$$f(x) = 3 \quad , \quad x > 3$$

The maximizing set for f is given by

$$\mu_M(x) = \frac{x + 2}{4} \quad , \quad 0 \leq x \leq 2$$

$$\mu_M(x) = \frac{6 - x}{4} \quad , \quad 2 < x \leq 3$$

$$\mu_M(x) = \frac{3}{4} \quad , \quad x > 3$$

Correspondingly, the fuzzy maximum of f is given by

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

$$= 0 \quad \text{elsewhere.}$$

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

(13) Minimizing set for f = Maximizing set for -f.

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

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

(15) $\mu_M(x) + \mu_m(x) = 1$ for non-definite f

and

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

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 which are

related to one another in this way are weakly complementary.) In particular, if f is non-definite, then by (15) M and m are complementary fuzzy subsets of X - a property which is in accord with our intuition.

The maximizing and minimizing sets of a reward function contain the type of information which is usually provided by a 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.

3. 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 [1], a fuzzy algorithm may be equated with a fuzzy Turing machine. In [4], 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 [5], [6]. 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 non-fuzzy 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

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

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

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

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

is equivalent to expressing L as the collection of ordered pairs

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

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

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

and

$$L' = \mu_1'\alpha_1' + \ldots + \mu_m'\alpha_m'$$

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

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

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

and

$$\alpha_i\alpha_j' = \text{concatenation of}\ \alpha_i \text{ and } \alpha_j'. \tag{20}$$

Example.

$$L = 0.3\ aa + 0.5\ aba$$

$$L' = 0.2\ ab + 0.9\ ba$$

Then

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

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

(21) $$\alpha \to \beta \text{ or } \alpha \to \beta \,.$$

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 which removes the first three occurrences of c from x. E.g., acabbcabcc is transformed into aabbabc.

A fuzzy version of this problem would be: Find a fuzzy Markoff algorithm which removes the first <u>few</u> occurrences of c from x. In this case, if we define <u>few</u> as the fuzzy set

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

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

FM(acabbcabcc) = 1.0 aabbcabcc + 1.0 aabbabcc + 0.8 aabbabc + 0.6 aabbab

More generally, let $\mathcal{F}(A^*)$ denote the set of all fuzzy subsets of A^* . Then a <u>fuzzy Markoff algorithm</u>, FM, may be regarded as a function from A^* to $\mathcal{F}(A^*)$ which satisfies certain conditions and is characterized

in a particular way which will be described presently.

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

$$(22) \qquad L = \mu_1\alpha_1 + \ldots + \mu_n\alpha_n \ , \ \alpha_i \in A^* \ , \quad i = 1, \ldots, n \ .$$

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

$$(23) \qquad FM(L) = FM(\mu_1\alpha_1 + \ldots + \mu_n\alpha_n)$$

$$= \mu_1 \, FM(\alpha_1) + \ldots + \mu_n \, FM(\alpha_n)$$

which implies that FM is a linear operator in A^*. In consequence of (23), 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 non-fuzzy Markoff algorithm, a typical production, P_i, in a fuzzy Markoff algorithm has the appearance

$$(24) \qquad P_i : \ \alpha \rightarrow \mu_1\beta_1 + \ldots + \mu_k\beta_k + \mu_{k+1} \, \beta_{k+1} \cdot + \ldots + \mu_n\beta_n \cdot$$

where the β's $\in A^*$, the μ's are numbers in the interval $[0,1]$, and the terms ending with a period are the <u>terminating</u> 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 $x = u \, \alpha \, v$, where $u, \alpha, v \in A^*$ and α is not a substring of u. (I.e., α in $u \, \alpha \, v$ 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

$$(25) \qquad u \, \alpha \, v = u \, (\mu_1\beta_1 + \ldots + \mu_k\beta_k + \mu_{k+1} \, \beta_{k+1} \cdot + \ldots + \mu_n\beta_n \cdot) v$$

$$= \mu_1 u\beta_1 v + \ldots + \mu_k u\beta_k v + \mu_{k+1} u\beta_{k+1} v \circ + \ldots + \mu_n u\beta_n v \circ$$

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

$$(26) \quad \rho\, u\, \alpha\, v = \rho\, \mu_1 u\beta_1 v + \ldots + \rho\, \mu_k u\beta_k v + \rho\, \mu_{k+1} u\beta_k v \circ + \ldots + \rho\, \mu_n u\beta_n v \circ$$

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

Example. Suppose x = abbabbb and

ba → 0.3 ab + 0.9 a ∘

Then

abbabbb = 0.3 ababbbb + 0.9 ababbb∘

and

0.4 abbabbb = 0.3 ababbbb + 0.4 ababbb∘

In summary, a production P_i is applicable to a string x if its antecedent is a substring of x. The result of applying of P_i to x is expressed by (25) and (26). (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 (23) and the rewriting rules (25) and (26).

Definition of a fuzzy Markoff algorithm

A fuzzy Markoff algorithm, FM, is a function from A^* to $\mathcal{F}(A^*)$ which satisfies (23) and is characterized by (a) a finite alphabet A; (b) possibly a finite auxiliary alphabet A' (comprising various markers which may be needed for bookkeeping purposes); and (c) a finite sequence of productions $P_1, \ldots, P_n$ of the form (24), with P_n: $\Lambda \rightarrow \Lambda$∘ ($\Lambda$ = null string).

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

$$L = \mu_1 v_1 + \ldots + \mu_r v_r \quad , \quad v_i \in A^* \quad , \quad i = 1, \ldots, r$$

then, as in (23)

$$\overline{FM}(L) = \mu_1 \overline{FM}(v_1) + \ldots + \mu_r \overline{FM}(v_r) \tag{27}$$

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

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

$$D(v_i) = \text{sum of terms in } \overline{FM}(v_i) \text{ which terminate in a period} \tag{28}$$

$$B(v_i) = \text{sum of the remaining terms in } \overline{FM}(v_i) \tag{29}$$

and

$$D(L) = \mu_1 D(v_1) + \ldots + \mu_r D(v_r) \tag{30}$$

$$B(L) = \mu_1 B(v_1) + \ldots + \mu_r B(v_r) \tag{31}$$

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

$$\overline{FM}(L) = D(L) + B(L) \tag{32}$$

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

1. Set L = x = initial string
2. Set FM(x) = θ (empty set)

3. Apply $\overline{FM}$ to L and compute $\overline{FM}(L) = B(L) + D(L)$.

4. Set

(33) $FM(x) = FM(x) + D(L)$

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: ab → 0.3 bb + 0.6 a + 0.9 ab.

P_2: ba → 0.8 a. + 1.0 bb.

P_3: b → 0.6 a.

Λ → 1.0 Λ.

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

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

D(L) = 0.9 abba.

B(L) = 0.3 bbba + 0.6 aba

and

FM(x) = 0.9 abba.

Since B(L) is non-empty, we apply $\overline{FM}$ to L = B(L) yielding

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

Now, applying P_2 to bbba we get

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

and applying P_1 to aba we have

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

Thus

$$FM(L) = 0.3\ bba + 0.6\ aa + 0.6\ aba. + 0.3\ bba. + 0.3\ bbbb.$$

$$D(L) = 0.6\ aba. + 0.3\ bba. + 0.3\ bbbb.$$

$$B(L) = 0.3\ bba + 0.6\ aa$$

and

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

Since B(L) is non-empty, we apply $\overline{FM}$ to L = B(L), obtaining

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

Next, applying P_2 to bba, we get

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

and applying P_3 to aa, we obtain

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

thus

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

$$D(L) = 0.3\ ba. + 0.3\ bbb. + 0.6\ aa.$$

$$B(L) = \theta$$

and finally

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

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

The execution of a fuzzy Markoff algorithm may be likened to a birth-and-death process in which the operation with $\overline{FM}$ on a string v_i gives rise to the birth of new strings, represented by $B(v_i)$, and the

death of others, represented by $D(v_i)$. In the same sense, B(L) and D(L) represent the "newly born" and the "dead" strings resulting from operating with $\overline{FM}$ on a "live" fuzzy set of strings L. Finally, F(x) plays the role of the population of the dead in a cemetery, with (33) 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. L. A. Zadeh, "Fuzzy Algorithms," Information and Control, vol. 12, pp. 99-102, February 1968.
2. S. S. L. Chang and L. A. Zadeh, "Fuzzy Mapping and Control," Trans. on Systems, Man and Cybernetics, vol. SMC-2, pp. 30-34, January 1972.
3. L. A. Zadeh, "Fuzzy Sets," Information and Control, vol. 8, pp. 338-353, June 1965.
4. E. Santos, "Fuzzy Algorithms," Information and Control, vol. 17, pp. 326-339, November 1970.
5. A. T. Berztiss, "Data Structures," Mc Graw-Hill Book Co., New York, 1971.
6. H. Galler and A. Perlis, " A View of Programming Languages," Addison Wesley, Reading, Mass., 1970.

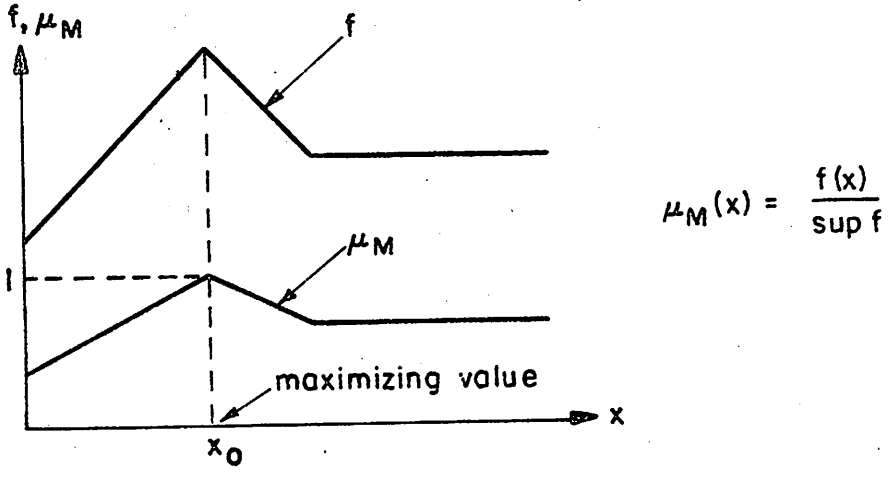

Fig. 1. The maximizing set for a positive-definite reward function.

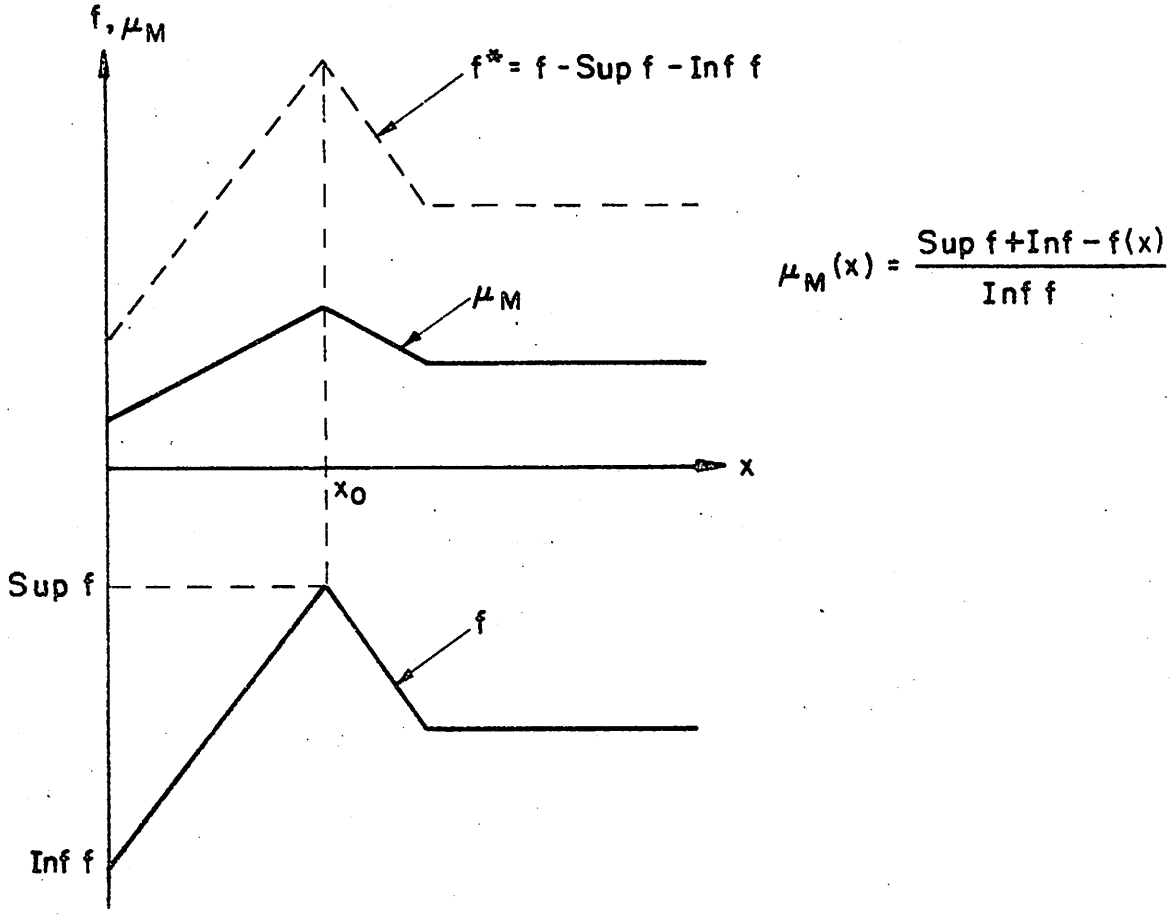

Fig. 2. The maximizing set for a non-definite reward function.

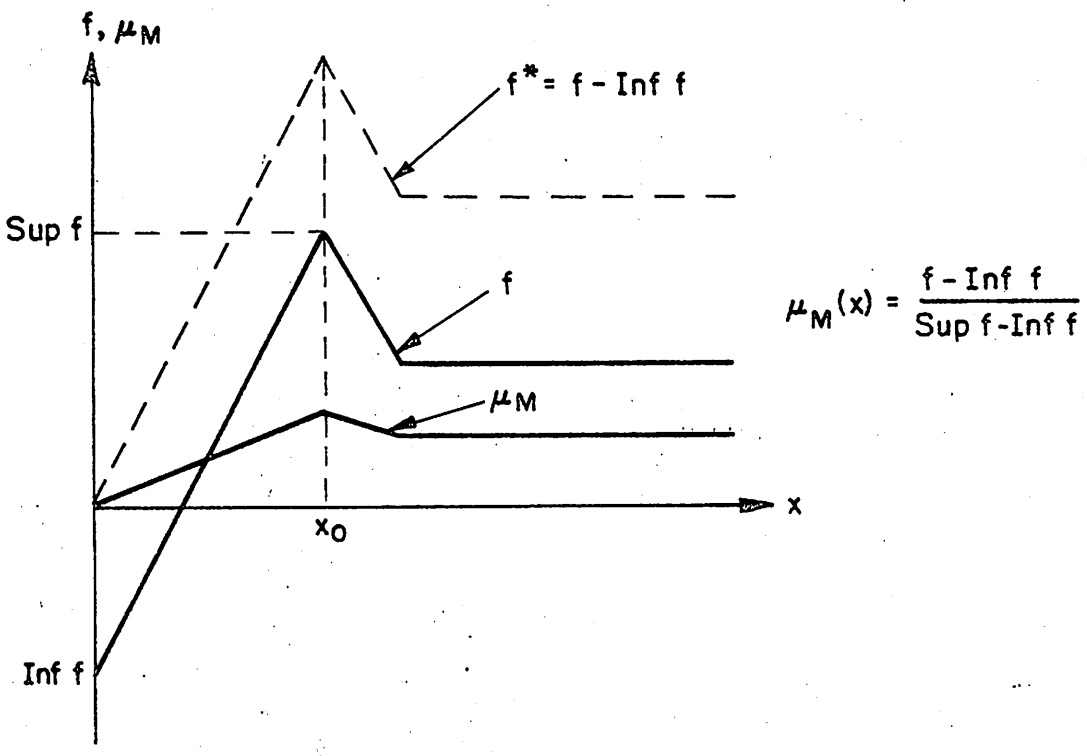

Fig. 3. The maximizing set for a negative-definite reward function.