Control and Cybernetics

vol. **25** (1996) No. 6

Multistage control under fuzziness using genetic algorithms

by

Janusz Kacprzyk
Systems Research Institute, Polish Academy of Sciences,
01-447 Warszawa, Newelska 6, Poland
E-mail: kacprzyk@ibspan.waw.pl

Abstract: We consider the classic Bellman and Zadeh's (1970) multistage control problem under fuzzy constraints imposed on controls applied and fuzzy goals imposed on states attained. The fuzzy decision, serving the purpose of a performance function, is the intersection of the fuzzy constraints and goals. An optimal sequence of controls is sought which maximizes the fuzzy decision over a fixed and specified planning horizon. The basic cases of deterministic and fuzzy systems under control are discussed. The use of a genetic algorithm is shown to be a viable alternative to the traditionally employed solution techniques: Bellman and Zadeh's (1970) dynamic programming, augmented with Kacprzyk's (1993a–c) interpolative reasoning, Kacprzyk's (1978a, 1979) branch-and-bound, and Francelin and Gomide's (1992, 1993) and Francelin, Gomide and Kacprzyk's (1995) neural-network-based approach.

Keywords: fuzzy control, multistage fuzzy control, fuzzy dynamic programming, branch-and-bound, neural network, genetic algorithm.

1. Introduction

In recent years we witness an unprecedented interest in fuzzy (logic) control, which is triggered and amplified by successful applications in both specialized areas (e.g., control of technological processes, cranes, elevators) and everyday products (e.g., washing machines, refrigerators, cameras).

Fuzzy (logic) control was introduced by Mamdani (1974), and its essence (even including newer approaches as, e.g., neuro-fuzzy control) may be summarized as follows:

- the model of the process under control is unknown or too expensive to derive, and we "do not care" about it;
- an experienced process operator knows how to well control the process, and this knowledge may be expresses by some linguistic IF–THEN rules,

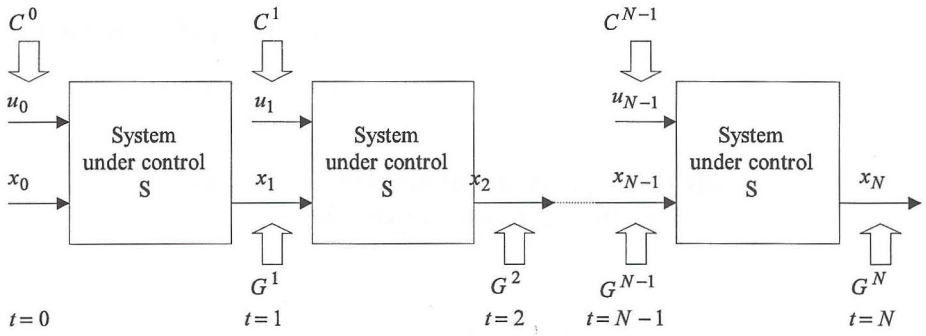


Figure 1. A general framework of the prescriptive, model-based approach to multistage fuzzy control

- the operator is, however, not sure what his or her performance function is, and if controls are the "best" (optimal).

Notice that this is clearly a *descriptive, non-model-based approach.*

The above may be viewed as contradicting the traditional control paradigm which may be outlined as follows:

- a (fuzzy, in general) model of the system under control is known,
- an explicit performance function is known,
- best (optimal) controls are sought (by an algorithm).

Notice that this is clearly a *prescriptive, model-based approach.*

Fortunately enough, such a prescriptive, model-based approach can also be devised for fuzzy control. In fact, it is even earlier as its roots are seminal Chang's (1969a, b) and Bellman and Zadeh's (1970) papers. For details we refer the interested reader to Kacprzyk's (1983a, 1997) books.

Basically, the general control framework adopted in those works, and also in this article, may be depicted as in Figure 1.

We start from an initial state at control stage (time) $t = 0$, x_0, apply a control at $t = 0$, u_0, attain a state at time $t = 1$, x_1, apply u_1, ... Finally, being at control stage $t = N-1$ in state x_{N-1} we apply control u_{N-1} and attain the final state x_N.

The dynamics of the system under control, S, is assumed known and given by state transitions from state x_t to x_{t+1} under control u_t, the consecutive controls applied u_t are subjected to fuzzy constraints C^t, and on the states attained x_{t+1} fuzzy goals G^{t+1} are imposed, $t = 0, 1, \ldots, N-1$.

The performance (goodness) of the control process is gauged by some (aggregation) measure of how well, at all the consecutive control stages, the fuzzy constraints on controls and fuzzy goals on states are satisfied. And an optimal sequence of controls at the consecutive control stages, $u_0^*, \ldots, u_{N-1}^*$, is sought (to be determined by an algorithm).

Notice that in this framework we proceed in a sequential manner, at discrete

control stages, and hence the problem considered is termed multistage fuzzy control.

It can be seen that the above general scheme of a prescriptive approach to multistage fuzzy control may be viewed to give rise to the following general *problem classes*, for the following types of the termination time and system under control:

- type of *termination time*:
  1. fixed and specified in advance,
  2. explicitly specified (as the moment of entering for the first time a termination set of states),
  3. fuzzy, and
  4. infinite;
- type of *system under control*:
  1. deterministic,
  2. stochastic, and
  3. fuzzy.

For the solution of the above problem classes, a variety of techniques has been proposed, mostly of dynamic programming and branch-and-bound type, see Kacprzyk (1997).

In this paper we will consider the basic case of a fixed and specified termination time, and the deterministic and fuzzy systems under control. We will indicate conceptual and numerical difficulties in the use of the above mentioned traditional solution techniques, and propose the use of a genetic algorithm which is conceptually simple, and proves to be numerically efficient.

In Section 2 we will briefly present Bellman and Zadeh's (1970) general approach to decision making under fuzziness which will be employed as a framework. In Section 3 we will outline how to use Bellman and Zadeh's (1970) approach to formulate multistage fuzzy control problems. In Section 4 we will first consider multistage fuzzy control with a deterministic system, and then show how a genetic algorithm can be employed for solution. In Section 5 we will first consider multistage fuzzy control with a fuzzy system, and then show how a genetic algorithm can be employed for solution. In Section 6 we provide some concluding remarks, and then an extensive list of literature.

Our fuzzy-sets-related notation will be standard. A fuzzy set A in $X = \{x\}$ will be characterized (and practically equated with its membership function $\mu_A : X \longrightarrow [0,1]$ such that $\mu_A(x) \in [0,1]$ is the grade of membership of element $x \in X$ in fuzzy set A. The operations on fuzzy sets will be assumed standard, in particular the intersection of A and B in X will be defined as $\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x) = \min[\mu_A(x), \mu_B(x)]$, for each $x \in X$. Moreover, the use of other operations, notably t-norms, will be mentioned. Details on these and other related aspects can be found in any book on fuzzy sets theory such as Klir and

Yuan (1995). However, an initial chapter on fuzzy sets theory in Kacprzyk's (1997) book may better serve this purpose as discussions are there tailored to the specifics of multistage fuzzy control.

2. Multistage fuzzy control in Bellman and Zadeh's setting

In this section we will provide the reader with a brief introduction to Bellman and Zadeh's (1970) general approach to decision making under fuzziness, originally termed *decision making in a fuzzy environment*, a simple yet extremely powerful framework within which virtually all fuzzy models related to decision making, optimization and (optimal) control have been dealt with. This framework will also be employed in our next discussions.

2.1. Decision making in a fuzzy environment in Bellman and Zadeh's setting

In Bellman and Zadeh's (1970) setting the imprecision (fuzziness) of the environment within which decision making (or control) proceeds is modeled by the so-called *fuzzy environment* which consists of fuzzy goals, fuzzy constraints, and a fuzzy decision.

We start with the assumption of some set of possible (feasible, relevant, ...) *options* (alternatives, variants, choices, decisions, ...) denoted by $X = \{x\}$.

The *fuzzy goal* is now defined as a fuzzy set G in the set of options X, characterized by its membership function $\mu_G : X \longrightarrow [0,1]$ such that $\mu_G(x) \in [0,1]$ specifies the grade of membership of a particular option $x \in X$ in the fuzzy goal G.

The *fuzzy constraint* is similarly defined as a fuzzy set C in the set of options X, characterized by its membership function $\mu_C : X \longrightarrow [0,1]$ such that $\mu_C(x) \in [0,1]$ specifies the grade of membership of a particular option $x \in X$ in the fuzzy constraint C.

For example, suppose that $X = R$, the set of real numbers. Then the fuzzy goal "x should be *much larger* than 5" may be represented by a fuzzy set whose membership function, $\mu_G(x)$, is shown in Figure 2. On the other hand, the fuzzy constraint "x should be *more or less* 6" may be represented by a fuzzy set whose membership function, $\mu_C(x)$, is also shown in Figure 2.

An important issue is how the fuzzy goal and fuzzy constraint are to be interpreted. On the one hand, if we suppose that $f : X \longrightarrow R$ is a conventional performance (objective) function which associates with each $x \in X$ a real number $f(x) \in R$, and if we denote $M = \max_{x \in X} f(x)$, assuming $M \leq \infty$, then $\mu_G(x)$ can be defined as a normalized performance function f, i.e.

$$\mu_G(x) = \frac{f(x)}{M} = \frac{f(x)}{\max_{x \in X} f(x)}, \qquad \text{for each } x \in X \tag{1}$$

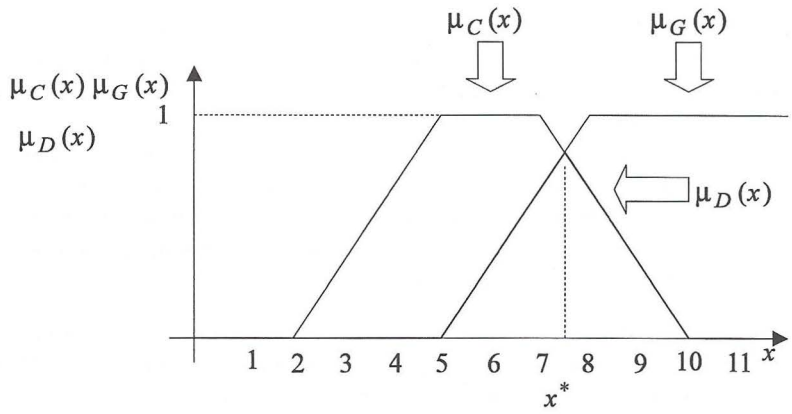


Figure 2. Fuzzy goal, fuzzy constraint, fuzzy decision, and the optimal (maximizing) decision

On the other hand, a fuzzy goal may be viewed from a different perspective, which is presumably more convenient for our discussions, in terms of *satisfaction levels*. The piecewise linear membership function of G in Figure 2 should be understood as: if the value of x attained is at least $\overline{x}_G$ (equal 8), which is the *satisfaction level* of x, then $\mu_G(x) = 1$ which means that we are fully satisfied with x. If the x attained does not exceed $\underline{x}_G$ $(= 5)$, which is the lowest possible value of x, then $\mu_G(x) = 0$ which means that we are fully dissatisfied with such a value of x. For the intermediate values, $\underline{x}_G < x < \overline{x}_G$, we have $0 < \mu_G(x) < 1$ which means that our satisfaction as to a particular value of x is intermediate. The meaning of C is analogous. The above interpretation provides a "common denominator" for the fuzzy goal and fuzzy constraint which may be treated in an analogous way which is one of merits of Bellman and Zadeh's (1970) approach.

Now the following general formulation of the decision making problem in a fuzzy environment may be postulated:

"Attain G and satisfy C" (2)

The fuzziness of the fuzzy goal and fuzzy constraint implies the fuzziness of the outcome (decision), which is called a fuzzy decision, and is a result of some aggregation of the fuzzy goal and fuzzy constraint which is equivalent to the intersection of two fuzzy sets that corresponds to the "and" connective.

Formally, if G is a fuzzy goal and C is a fuzzy constraint, both defined as fuzzy sets in $X = \{x\}$, the *fuzzy decision* D is a fuzzy set defined also in X such that

$$\mu_D(x) = \mu_G(x) \wedge \mu_C(x), \qquad \text{for each } x \in X \tag{3}$$

where "$\wedge$" is the minimum operation, i.e. $a \wedge b = \min(a, b)$.

Example 1 Suppose that G is "x should be much larger than 5", and C is "x should be more or less 6", as in Figure 2. The membership function of fuzzy decision is given in bold line, and should be interpreted as follows. The set of possible options is the interval $[5, 10]$ because $\mu_D(x) > 0$, for $5 \leq x \leq 10$. The other options, i.e. $x < 5$ and $x > 10$ are impossible since $\mu_D(x) = 0$. The value of $\mu_D(x) \in [0, 1]$ may be meant as the degree of satisfaction from the choice of a particular $x \in X$, from 0 for full dissatisfaction (impossibility of x) to 1 for full satisfaction, through all intermediate values; thus, the higher the value of $\mu_D(x)$, the higher the satisfaction from x. □

Note that in Figure 2, $\mu_D(x) < 1$ which means that there is no option which fully satisfies both the fuzzy goal and fuzzy constraint. In other words, there is a discrepancy or conflict between the goal and constraint.

The fuzzy decision provides a fuzzy solution to the decision making problem (2). In practice, however, if we wish to implement such a solution, we need to find a nonfuzzy solution. A straightforward choice is here the one corresponding to the highest value of $\mu_D(x)$.

The *maximizing decision* is defined as an $x^* \in X$ such that

$$\mu_D(x^*) = \max_{x \in X} \mu_D(x) \tag{4}$$

and an example may be found in Figure 2 where $x^* = 7.5$.

The fuzzy decision (3), the so-called min–type fuzzy decison, is the most widely used, but one can use other aggregation operators, notably t-norms; details can be found in Kacprzyk's (1997) book. In the following we will employ the min–type fuzzy decision only, tacitly assuming that our discussions will be extendable (in most cases) to other types of fuzzy decision.

The case of one fuzzy goal and fuzzy constraint is very illustrative, but in all nontrivial practical problems we face multiple fuzzy goals and fuzzy constraints.

Suppose that we have $n > 1$ fuzzy goals, $G_1, \ldots, G_n$, and $m > 1$ fuzzy constraints, $C_1, \ldots, C_m$, all defined in X.

The fuzzy decision can be defined analogously as in the case of one fuzzy goal and one fuzzy constraint, that is

$$\begin{aligned} \mu_D(x) = {} & \mu_{G_1}(x) \wedge \ldots \mu_{G_n}(x) \wedge \\ & \wedge \mu_{C_1}(x) \wedge \ldots \wedge \mu_{C_m}(x), \qquad \text{for each } x \in X \end{aligned} \tag{5}$$

The maximizing decision $x^* \in X$ is found as in (4), i.e.

$$\mu_D(x^*) = \max_{x \in X} \mu_D(x).$$

Our discussion of the Bellman and Zadeh's (1970) approach has concerned so far the fuzzy goals and fuzzy constraints defined in the same space X. However, for the issues considered here, and for virtually all applications in general, an extension of the approach is needed to cover the case of fuzzy goals and fuzzy constraints defined as fuzzy sets in different spaces.

Suppose that the fuzzy constraint C is defined as a fuzzy set in $X = \{x\}$, and the fuzzy goal G is defined as a fuzzy set in $Y = \{y\}$. Moreover, suppose that a function $f : X \longrightarrow Y$, $y = f(x)$, is known. Typically, X and Y may be sets of options and outcomes, causes and effects, etc.

The *induced fuzzy goal* G' in X generated by the given fuzzy goal G in Y is defined as

$$\mu_{G'}(x) = \mu_G[f(x)], \qquad \text{for each } x \in X \tag{6}$$

Example 2 Let $X = \{1, 2, 3, 4\}$, $Y = \{2, 3, \ldots, 10\}$, and $y = 2x + 1$. If now

$$G = 0.1/2 + 0.2/3 + 0.4/4 + 0.5/5 + 0.6/6 + 0.7/7 + 0.8/8 + 1/9 + 1/10$$

then

$$G' = \mu_G(3)/1 + \mu_G(5)/2 + \mu_G(7)/7 = 0.2/1 + 0.5/2 + 0.7/3 + 1/4$$

□

The *fuzzy decision* is now defined analogously, i.e. as

$$\mu_D(x) = \mu_{G'}(x) \wedge \mu_C(x), \qquad \text{for each } x \in X \tag{7}$$

Finally, for $n > 1$ fuzzy goals $G_1, \ldots, G_n$ defined in Y, $m > 1$ fuzzy constraints $C_1, \ldots, C_m$ defined in X, and a function $f : X \longrightarrow Y$, $y = f(x)$, we have

$$\begin{aligned} \mu_D(x) = {} & \mu_{G'_1}(x) \wedge \cdots \wedge \mu_{G'_n}(x) \\ & \wedge \mu_{C_1}(x) \wedge \cdots \wedge \mu_{C_n}(x), \qquad \text{for each } x \in X \end{aligned} \tag{8}$$

In all the types of fuzzy decision, an optimal decision is assumed, analogously as in the case of one fuzzy goal and one fuzzy constraint, to be the *maximizing decision* defined as (4), i.e. $\mu_D(x^*) = \max_{x \in X} \mu_D(x)$.

We are now in a position to proceed to the formulation of multistage fuzzy control problems in Bellman and Zadeh's (1970) setting presented in this section.

3. Multistage fuzzy control in Bellman and Zadeh's setting

Now, we will apply the general Bellman and Zadeh's (1970) approach presented in Section 2 to formulate multistage fuzzy control problems. Decisions will be referred to as *controls*, the discrete time moments at which decisions are to be made – as *control stages*, and the input–output (or cause–effect) relationship – as a *system under control*.

We will start with the basic, simpler case of a deterministic system under control, and then proceed to the case of a fuzzy system.

Suppose that the control space is $U = \{u\} = \{c_1, \ldots, c_m\}$ and the state space is $X = \{x\} = \{s_1, \ldots, s_n\}$, and both are assumed finite. For simplicity, the control is equated with the input, and the state with the output.

The *control process* proceeds basically as it has already been depicted in Figure 1. In the beginning we are in an initial state $x_0 \in X$. We apply a control $u_0 \in U$ which is subjected to a fuzzy constraint $\mu_{C^0}(u_0)$. We attain a state $x_1 \in X$ via a known input–output (cause–effect) relationship, i.e. a state transition equation of the system under control S; a fuzzy goal $\mu_{G^1}(x_1)$ is imposed on x_1. Next, we apply a control u_1 which is subjected to a fuzzy constraint $\mu_{C^1}(u_1)$, and attain a fuzzy state x_2 on which a fuzzy goal $\mu_{G^2}(x_2)$ is imposed, etc.

The states and state transition equations (i.e. the system under control) can be deterministic, stochastic and fuzzy. In this paper we assume the basic cases of a deterministic and fuzzy systems under control. The formulation and analysis of the control process for each of them is different, and will now be consecutively discussed.

3.1. The case of a deterministic system under control

Suppose that the system under control is deterministic and its temporal evolution is governed by a *state transition equation*

$$x_{t+1} = f(x_t, u_t), \qquad t = 0, 1, \ldots \tag{9}$$

where $x_t, x_{t+1} \in X = \{s_1, \ldots, s_n\}$ are the states at control stages t and $t+1$, respectively, and $u_t \in U = \{c_1, \ldots, c_m\}$ is the control at control stage t.

At each control stage t, $t = 0, 1, \ldots$, the control applied $u_t \in U$ is subjected to a *fuzzy constraint* $\mu_{C^t}(u_t)$, and on the state attained $x_{t+1} \in X$ a *fuzzy goal* is imposed.

The *initial state* is $x_0 \in X$ and is assumed to be known, and given in advance. The *termination time* (planning, or control, horizon), i.e. the maximum number of control stages, is denoted by $N \in \{1, 2, \ldots\}$, and is assumed to be fixed and specified in advance throughout this paper.

The *performance* (goodness) of the multistage fuzzy control process is evaluated by the fuzzy decision

$$\begin{aligned} &\mu_D(u_0, \ldots, u_{N-1} \mid x_0) = \\ &\quad = \mu_{C^0}(u_0) \wedge \mu_{G^1}(x_1) \wedge \cdots \wedge \mu_{C^{N-1}}(u_{N-1}) \wedge \mu_{G^N}(x_N) \end{aligned} \tag{10}$$

In most cases, however, a slightly simplified form of the fuzzy decision is used by assuming that all the subsequent fuzzy controls, $u_0, u_1, \ldots, u_{N-1}$, are subjected to the fuzzy constraints, $\mu_{C^0}(u_0), \mu_{C^1}(u_1), \ldots, \mu_{C^{N-1}}(u_{N-1})$, while the fuzzy goal is just imposed on the final state x_N, $\mu_{G^N}(x_N)$. In such a case the fuzzy decision becomes

$$\mu_D(u_0, \ldots, u_{N-1} \mid x_0) = \mu_{C^0}(u_0) \wedge \cdots \wedge \mu_{C^{N-1}}(u_{N-1}) \wedge \mu_{G^N}(x_N) \tag{11}$$

We will basically assume such a simplified form of the fuzzy decision in the following, but this will not limit the generality of discussion.

The multistage control problem in a fuzzy environment is now formulated as to find an optimal sequence of controls $u_0^*, \ldots, u_{N-1}^*$, $u_t^* \in U$, $t = 0, 1, \ldots, N-1$, such that

$$\mu_D(u_0^*, \ldots, u_{N-1}^* \mid x_0) = \max_{u_0, \ldots, u_{N-1} \in U} \mu_D(u_0, \ldots, u_{N-1} \mid x_0) \tag{12}$$

This problem can be solved using the following two basic traditional techniques:

- dynamic programming, and
- branch-and-bound,

and also using the following two new ones:

- a neural network, and
- a genetic algorithm.

In this section we will outline the use of the three first techniques, and it will be obvious that they suffer from numerical problems (the infamous curse of dimensionality) caused by a combinatorial character of the problem considered.

In Section 4 we will present the solution by using a genetic algorithm.

3.1.1. Solution by dynamic programming

The application of dynamic programming for the solution of problem (12) was proposed in the seminal paper of Bellman and Zadeh (1970).

First, let us slightly rewrite (12) as to find $u_0^*, \ldots, u_{N-1}^*$ such that

$$\mu_D(u_0^*, \ldots, u_{N-1} \mid x_0) = \max_{u_0, \ldots, u_{N-1}} [\mu_{C^0}(u_0) \wedge \ldots$$
$$\ldots \wedge \mu_{C^{N-1}}(u_{N-1}) \wedge \mu_{G^N}(f(x_{N-1}, u_{N-1}))] \tag{13}$$

Clearly, its structure makes the application of dynamic programming possible. Namely, the last two right-hand-side terms, i.e.

$$\mu_{C^{N-1}}(u_{N-1}) \wedge \mu_{G^N}(f(x_{N-1}, u_{N-1}))$$

depend only on control u_{N-1} and not on any previous controls, and hence the maximization over $u_0, \ldots, u_{N-1}$ in (13) can be divided into:

- maximization over $u_0, \ldots, u_{N-2}$, and
- maximization over u_{N-1},

that is

$$\mu_D(u_0^*, \ldots, u_{N-1}^* \mid x_0) =$$
$$= \max_{u_0, \ldots, u_{N-2}} \{\mu_{C^0}(u_0) \wedge \ldots \wedge \mu_{C^{N-2}}(u_{N-2}) \wedge$$
$$\wedge \max_{u_{N-1}} [\mu_{C^{N-1}}(u_{N-1}) \wedge \mu_{G^N}(f(x_{N-1}, u_{N-1}))]\} \tag{14}$$

And further, continuing the same line of reasoning, for u_{N-2}, u_{N-3}, etc. we arrive at the following set of dynamic programming recurrence equations:

$$\begin{cases} \mu_{G^{N-i}}(x_{N-i}) = \max_{u_{N-i}}[\mu_{C^{N-i}}(u_{N-i}) \wedge \mu_{G^{N-i+1}}(x_{N-i+1})] \\ x_{N-i+1} = \mathrm{f}(x_{N-i}, u_{N-i}), \qquad i = 0, 1, \ldots, N \end{cases} \tag{15}$$

where $\mu_{G^{N-i}}(x_{N-i})$ may be regarded as a fuzzy goal at control stage $t = N - i$ induced by the fuzzy goal at $t = N - i + 1$, $i = 0, 1, \ldots, N$.

The optimal sequence of control sought, $u_0^*, \ldots, u_{N-1}^*$, is given by the successive maximizing values of u_{N-i}, $i = 1, \ldots, N$ in (15). Each such a maximizing value, u_{N-i}^* is obviously obtained as a function of x_{N-i}, i.e. a *policy*.

Example 3 Suppose that $X = \{s_1, s_2, s_3\}$, $U = \{c_1, c_2\}$, $N = 2$, and fuzzy constraints and fuzzy goal are

$$\begin{aligned} C^0 &= 0.7/c_1 + 1/c_2 \\ C^1 &= 1/c_1 + 0.8/c_2 \\ G^2 &= 0.3/s_1 + 1/s_2 + 0.8/s_3 \end{aligned}$$

and the state transition equation (9) is given as

$$x_{t+1} = \begin{array}{r|ccc} & x_t = s_1 & s_2 & s_3 \\ \hline u_t = c_1 & s_1 & s_3 & s_1 \\ c_2 & s_2 & s_1 & s_3 \end{array} \tag{16}$$

First, using (15) for $i = 1$, we obtain $G^1 = 0.6/s_1 + 0.8/s_2 + 0.6/s_3$, and the corresponding optimal control policy

$$a_1^*(s_1) = c_2 \quad a_1^*(s_2) = c_1 \quad a_1^*(s_3) = c_2$$

Next, (15), for $i = 2$, yields $G^0 = 0.8/s_1 + 0.6/s_2 + 0.6/s_3$ and the corresponding optimal control policy

$$a_0^*(s_1) = c_2 \quad a_0^*(s_2) \in \{c_1, c_2\} \quad a_1^*(s_3) \in \{c_1, c_2\}$$

Therefore, for instance, if we start at $t = 0$ from $x_0 = s_1$, then $u_0^* = a_0^*(s_1) = c_2$ and we obtain $x_1 = s_2$. Next, at $t = 1$, $u_1^* = a_1^*(s_2) = c_1$ and $\mu_D(u_0^*, u_1^* \mid s_1) = \mu_D(c_2, c_1 \mid s_1) = 0.8$. □

3.1.2. Solution by branch-and-bound

As an alternative to dynamic programming for solving problem (12), a branch-and-bound approach was proposed by Kacprzyk (1978a).

The branch-and-bound procedure starts from the initial state x_0. We apply control u_0 and proceed to state x_1. Next, we apply u_1 and proceed to x_2, etc. Finally, being in x_{N-1}, we apply u_{N-1} and attain x_N.

This may be represented as a decison tree whose nodes are associated with the particular states obtained, and whose edges represent the controls applied.

To each path there corresponds some value of the fuzzy decision (11) [or (10)], and the problem is to find the one with the highest value.

The branch-and-bound procedure is basically an implicit enumeration scheme (equivalent to the traversing of a possibly small number of paths in the decison tree) the very essence of which boils down to the answering of the following question:

> If we currently (at the current control stage) arrive at some node (state), then to which node (out of those traversed so far) should we most rationally (to proceed further along the currently most promising path) add next edges (controls)?

To present the idea of Kacprzyk's (1978a) approach, let us first denote

$$\begin{cases} v_0 = \mu_{C^0}(u_0) \\ \dots \\ v_k = \mu_{C^0}(u_0) \wedge \dots \wedge \mu_{C^k}(u_k) = v_{k-1} \wedge \mu_{C^k}(u_k) \\ \dots \\ v_{N-1} = \mu_{C^0}(u_0) \wedge \dots \wedge \mu_{C^{N-1}}(u_{N-1}) = v_{N-2} \wedge \mu_{C^{N-1}}(u_{N-1}) \\ v_N = \mu_{C^0}(u_0) \wedge \dots \wedge \mu_{C^{N-1}}(u_{N-1}) \wedge \mu_{G^N}(x_N) = \\ \quad = v_{N-1} \wedge \mu_{G^N}(x_N) = \mu_D(u_0, \dots, u_{N-1} \mid x_0) \end{cases} \tag{17}$$

The use of "$\wedge$" (minimum) implies that if we consider some sequence of controls $u_0, \dots, u_k$, $0 < k < N-1$, then, for each $k < w \leq N-1$:

$$v_k \geq v_w = v_k \wedge \mu_{C^{k+1}}(u_{k+1}) \wedge \dots \wedge \mu_{C^w}(u_w) \tag{18}$$

because, due to "$\wedge$", by "adding" to v_k any further terms we cannot increase the value of v_w.

In particular, there also holds

$$v_k \geq v_N = \mu_D(u_0, \dots, u_{N-1} \mid x_0) \tag{19}$$

Now, if we are at the k-th control stage, and have traversed so far some nodes and edges (from x_0 to x_k), the most promising current choice is to choose the most promising node, i.e. the one which corresponds to the greatest value of v_i attained so far, $i-1, \dots, k$. The other nodes cannot lead (at that particular moment!) to any optimal solution since they cannot obviously yield any higher value of v_i if we add next edges.

The above property of the min-type fuzzy decision makes it possible to devise a branch-and-bound algorithm in which the branching is through the controls applied at the consecutive control stages, and the bounding is via the values of the particular v_k's, $k = 0, \dots, N$. For details, we refer the interested reader to Kacprzyk's (1983a, 1997) books.

Evidently, by solving Example 3 we obtain the same results as for dynamic programming.

3.1.3. Solution by an artificial neural network

This nonconventional solution techniques for solving problem (12) was proposed by Francelin and Gomide (1992, 1993), and Francelin, Gomide and Kacprzyk (1995).

Basically, we start with the dynamic programming formulation for solving problem (12) presented in Section 3.1.1.

First, we rewrite the dynamic programming recurrence equations (15) as

$$\begin{cases} \mu_{\overline{G}^{N-i}}(x_{N-i}) = \\ \quad = \underbrace{\max_{u_{N-i}} \underbrace{[\mu_{C^{N-i}}(u_{N-i}) \wedge \mu_{G^{N-i}}(x_{N-i}) \wedge \mu_{\overline{G}^{N-i+1}}(x_{N-i+1})]}_{\text{minimization at } t=N-i}}_{\text{maximization at } t=N-i} \\ x_{N-i+1} = f(x_{N-i}, u_{N-i}); \qquad i = 0, 1, \ldots, N \end{cases} \tag{20}$$

So, proceeding backwards from the final ($t = N$) to the initial ($t = 0$) control stage, at each particular control stage we perform two phases:

- minimization, and
- maximization

as schematically shown in (20).

Such a flow of computation "minimization at $t = N - 1$, maximization at $t = N - 1$, minimization at $t = N - 2$, maximization at $t = N - 2$, ..., minimization at $t = 0$, maximization at $t = 0$" may be modeled by a special neural network.

First, note that it cannot be a traditional neural network since we have here some "non-traditional" operations: the minimum "$\wedge$" and the maximization. We need some special types of neurons which may implement these two operations. Luckily enough, such neurons may be obtained as special cases of some generalized recurrent neurons proposed by Rocha (1993).

Francelin and Gomide's (1992, 1993) neural network for solving fuzzy dynamic programming problems is composed of alternate layers of min-type and max-type neurons [corresponding to the minimization and maximization phases indicated in (20)] of the type defined above.

The network's weights are not derived by training in usual manner, i.e. by feeding the network with examples, but are somehow designed, or even predetermined by the description of the problem (state transitions, fuzzy constraints and goals, etc.). So, from some points of view it may regarded as not a "real" neural network. However, on the other hand, it has a clear neural network topology, and – what is crucial for our purposes – it exhibits an inherent parallelism in its operation.

It is proved (Francelin and Gomide, 1992, 1993; Francelin, Gomide and Kacprzyk, 1995) that the above method yields the same results as dynamic programming presented in Section 3.1.1.

A detailed description of this approach is beyond the scope of our discussion, and the interested reader is referred to the source work of Francelin and Gomide (1992, 1993), or – even better – to Kacprzyk's (1997) book.

3.2. The case of a fuzzy system under control

In this section we will consider the case of a fuzzy system under control whose dynamics is given as a state transition equation

$$X_{t+1} = F(X_t, U_t), \qquad t = 0, 1, \ldots \tag{21}$$

where $X_t, X_{t+1} \in \mathcal{X}$ are fuzzy states at control stage t and $t+1$, respectively, and $U_t \in \mathcal{U}$ is a fuzzy control at control stage t, $t = 0, 1, \ldots$; $\mathcal{U} = \{C_1, \ldots, C_l\}$ is the set of fuzzy controls, and $\mathcal{X} = \{S_1, \ldots, S_q\}$ is the set of fuzzy states.

The state transition equation (21) may be equated with a conditioned fuzzy set whose membership function is $\mu_{X_{t+1}}(x_{t+1} \mid x_t, u_t)$, and then the state transitions are governed by (Kacprzyk, 1997):

$$\mu_{X_{t+1}}(x_{t+1}) = \max_{x_t \in X} [\mu_{X_t}(x_t) \wedge \mu_{X_{t+1}}(x_{t+1} \mid x_t, u_t)], \qquad \text{for each } x_{t+1} \in X \tag{22}$$

Notice also the the above general form of a state transition relation can be represented in various forms exemplified by a state transition equation itself, IF–THEN rules, a neural network, etc. This will not be considered here and we will refer the interested reader to Kacprzyk (1997). Moreover, we will not discuss the relevant problem of identification of fuzzy systems under control see, Cao and Rees (1992), Czogała and Pedrycz (1981, 1984), Kacprzyk (1997), Pedrycz (1993, 1996), Sugeno and Yasukawa (1993), etc.].

First, it should be noted that in case of the deterministic system under control, the consecutive controls applied, $u_0, \ldots, u_{N-1} \in U$, and the states attained, $x_1, \ldots, x_N \in X$, were nonfuzzy, hence we could directly determine their grade of membership in the fuzzy constraints, $\mu_{C^0}(u_0), \ldots, \mu_{C^{N-1}}(u_{N-1})$, and in the fuzzy goals, $\mu_{G^1}(x_1), \ldots, \mu_{G^N}(x_N)$, respectively (see Section 2).

In the case of a fuzzy system the control applied and states attained are fuzzy, and their grade of membership in the fuzzy constraints and in the fuzzy goal cannot be directly determined, and some manipulation ("trickery") is needed which will be employed below.

Suppose that at each t, $U_t \in \mathcal{U}$ is subjected to a fuzzy constraint $\mu_{C^t}(u_t)$, and on the resulting $X_{t+1} \in \mathcal{X}$ a fuzzy goal $\mu_{G^{t+1}}(x_{t+1})$ is imposed, $t = 0, 1, \ldots, N-1$. To account for the fuzziness of the controls and states, some redefinition of the fuzzy constraints and fuzzy goals is needed, for instance as follows:

$$\mu_{\overline{C}^t}(U_t) = 1 - diss(C^t, U_t), \qquad t = 0, 1, \ldots, N-1 \tag{23}$$

and

$$\mu_{\overline{G}^{t+1}}(U_{t+1}) = 1 - diss(G^{t+1}, U_{t+1}), \qquad t = 0, 1, \ldots, N-1 \tag{24}$$

where $diss : [0,1] \times [0,1] \longrightarrow [0,1]$ is some measure of dissemblance.

Traditionally, this measure is assumed to be a normalized distance between fuzzy sets, $d : \mathcal{X} \times \mathcal{X} \longrightarrow [0,1]$, so that (23) becomes

$$\mu_{\overline{C}^t}(U_t) = 1 - d(U_t, C^t), \qquad t = 0, 1, \ldots, N-1 \tag{25}$$

which will obviously serve the purpose of measuring the closeness (similarity) of U_t's and C^t's, and may be used instead of $\mu_{C^t}(u_t)$'s in the control problem formulation.

And similarly for the fuzzy goals: employing the same line of reasoning, we obtain

$$\mu_{\overline{G}^{t+1}}(X_{t+1}) = 1 - d(X_{t+1}, G^{t+1}), \qquad t = 0, 1, \ldots, N-1 \tag{26}$$

For the normalized distances, we usually employ:

- the normalized linear (Hamming) distance

$$d_l(X_N, G^N) = \frac{1}{N} \sum_{i=1}^{N} \mid \mu_{X_N}(s_i) - \mu_{G^N}(s_i) \mid \tag{27}$$

- the normalized quadratic (Euclidean) distance

$$d_q(X_N, G^N) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} [\mu_{X_N}(s_i) - \mu_{G^N}(s_i)]^2} \tag{28}$$

As to other choices, a degree of equality of two fuzzy sets proposed by Kacprzyk and Staniewski (1982) is also a plausible choice.

We may also use other indices or measures of (dis)similiarity, and one of them – Kaufmann and Gupta's (1985) dissimilarity index – will be discussed in Section 5.

Then, generally, the fuzzy decision is

$$\begin{aligned} &\mu_D(U_0, \ldots, U_{N-1} \mid X_0) = \\ &\quad = \mu_{\overline{C}^0}(U_0) \wedge \mu_{\overline{G}^1}(X_1) \wedge \ldots \wedge \mu_{\overline{C}^{N-1}}(U_{N-1}) \wedge \mu_{\overline{G}^N}(X_N) \end{aligned} \tag{29}$$

and we seek an optimal sequence of fuzzy controls $U_0^*, \ldots, U_{N-1}^*$ such that

$$\mu_D(U_0^*, \ldots, U_{N-1}^* \mid X_0) = \max_{U_0, \ldots, U_{N-1}} \mu_D(U_0, \ldots, U_{N-1} \mid X_0) \tag{30}$$

One can clearly readily obtain a simpler formulation with fuzzy constraints on all intermediate stages and a fuzzy goals at the end.

Its is easy to see that problem (30) is more complicated than problem (12) for the deterministic system under control. We will outline the application of more traditional dynamic programming and branch-and-bound approaches to solve the resulting control problems, and then a newer one based on interpolative reasoning. All will suffer from conceptual and/or numerical difficulties. In Section 5 we will propose a genetic algorithm and show its simplicity and efficiency.

3.2.1. Solution by dynamic programming

The application of dynamic programming to solving the problem of multistage fuzzy control of a fuzzy system under control was proposed by Baldwin and Pilsworth (1982).

The fuzzy system under control is assumed, as previously mentioned, to be described by a fuzzy state transition equation (21). At each t, U_t is subjected to $\mu_{C^t}(u_t)$, and on the resulting X_{t+1}, $\mu_{G^{t+1}}(x_{t+1})$ is imposed, $t = 0, 1, \ldots, N-1$.

Both the control at stage t, U_t, and the state at $t+1$, X_{t+1}, are now fuzzy, and hence their grades of membership in the fuzzy constraints C^t and G^{t+1} cannot be directly determined as the values of $\mu_{C^t}(u_t)$ and $\mu_{G^{t+1}}(x_{t+1})$, respectively. For each t, we construct a fuzzy relation R defined in $U \times X$ such that

$$\begin{aligned} &\mu_{R^t}(u_t, x_t) = \\ &\quad = \mu_{C^t}(u_t) \wedge \mu_{G^{t+1}}(x_{t+1}), \qquad \text{for each } u_t \in U, x_{t+1} \in X \end{aligned} \tag{31}$$

which represents the degree of how well (to which degree, between 0 and 1) C^t and G^{t+1} are satisfied.

The degree to which a particular U_t and X_{t+1} satisfy C^t and G^{t+1}, respectively, is defined as

$$\begin{aligned} T(U_t, R^t, X_{t+1}) &= \max_{x_{t+1} \in X} [\max_{u_t \in U} (\mu_{U_t}(u_t) \wedge \mu_{R^t}(u_t, x_t)) \wedge \mu_{X_{t+1}}(x_{t+1})] = \\ &= \max_{x_{t+1} \in X} [\max_{u_t} (\mu_{U_t}(u_t) \wedge \mu_{C^t}(u_t) \wedge \mu_{G^{t+1}}(x_{t+1}) \wedge \mu_{X_{t+1}}(x_{t+1}))] = \\ &= \max_{u_t \in U} [\mu_{U_t}(u_t) \wedge \mu_{C^t}(u_t)] \wedge \max_{x_{t+1} \in X} [\mu_{X_{t+1}}(x_{t+1}) \wedge \mu_{G^{t+1}}(x_{t+1})] \end{aligned} \tag{32}$$

The fuzzy decision is given as

$$\begin{aligned} &\mu_D(U_0, \ldots, U_{N-1} \mid X_0) = \\ &\quad = T(U_0, R^0, X_1) \wedge \ldots \wedge T(U_{N-1}, R^{N-1}, X_N) \end{aligned} \tag{33}$$

The problem, for a simpler case with fuzzy constraints at $t = 0, 1, \ldots, N-1$ and a fuzzy goal at $t = N$, is to determine $U_0^*, \ldots, U_{N-1}^*$ such that

$$\begin{aligned} &\mu_D(u_0^*, \ldots, U_{N-1}^* \mid X_0) = \\ &\quad = \max_{U_0, \ldots, U_{N-1}} \max_{u_0 \in U} [\mu_{U_0}(u_0) \wedge \mu_{C^0}(u_0) \wedge \ldots \wedge \max_{U_{N-1} \in U} (\mu_{U_{N-1}}(u_{N-1}) \wedge \\ &\qquad \wedge \mu_{C^{N-1}}(u_{N-1}) \wedge \max_{x_N} (\mu_{X_N}(x_N) \wedge \mu_{G^N}(x_N)) \ldots] \end{aligned} \tag{34}$$

It is now easy to see that the structure of (34) makes the use of dynamic programming possible, and the following set of dynamic programming recurrence

equations is obtained:

$$\begin{cases} \mu_{\overline{G}^N}(X_N) = \max_{x_N \in X}[\mu_{X_N}(x_N) \wedge \mu_{G^N}(x_N)] \\ \mu_{\overline{G}^{N-i}}(X_{N-i}) = \max_{U_{N-i} \in \mathcal{U}}[\max_{u_{N-i} \in U}(\mu_{U_{N-i}}(u_{N-i}) \wedge \\ \qquad \wedge \mu_{C^{N-i}}(u_{N-i})) \wedge \mu_{\overline{G}^{N-i+1}}(x_{N-i+1})] \\ \mu_{X_{N-i+1}}(x_{N-i+1}) = \max_{x_{N-i} \in X}[\max_{u_{N-i} \in U}(\mu_{U_{N-i}}(u_{N-i}) \wedge \\ \qquad \wedge \mu_{X_{N-i+1}}(x_{N-i+1} \mid x_{N-i}, u_{N-i})) \wedge \mu_{X_{N-i}}(x_{N-i})] \\ i = 1, \ldots N \end{cases} \tag{35}$$

In principle, the above set of dynamic programming recurrence equations may be solved. However, a serious difficulty may be seen just at first glance. First, $\mu_{\overline{G}^{N-i}}(X_{N-i})$ is to be specified for each possible fuzzy state $X_{N-i} \in \mathcal{X}$. Second, the maximization of $\mu_{U_{N-i}}(.)$ is to proceed over all (well, maybe not all but a large subset of) the fuzzy controls $U_{N-i} \in \mathcal{U}$. Evidently, the number of all the possible fuzzy controls U_{N-i} and fuzzy states X_{N-i} may be very high: infinite in the general case but at least very high in our context as we consider fuzzy sets defined in finite universes of discourse. This clearly makes the solution of (35) practically impossible.

Basically, the essence of Baldwin and Pilsworth's (1982) approach – it is also the same as Kacprzyk and Staniewski's (1982) approach – is to assume some relatively low number of standard (reference) fuzzy states and controls, perform the solution process in terms of them, and finally adjust the solution to reveal the "real" solution. For details we refer the interested reader to the source work (Baldwin and Pilsworth, 1982) or to Kacprzyk's (1997) book.

This concludes our glimpse at Baldwin and Pilsworth's (1982) dynamic-programming-based approach which is, unfortunately, very complicated and difficult to implement.

Now we will proceed to a conceptually and computationally simpler Kacprzyk's (1979) branch-and-bound approach, which appeared even earlier.

3.2.2. Solution by branch-and-bound

A branch-and-bound approach for solving problem (30) was proposed by Kacprzyk (1979). It is analogous to that discussed in Section 3.1.2. for the deterministic system under control.

Suppose that the fuzzy system under control is given by a fuzzy state transition equation (21), i.e. $X_{t+1} = F(X_t, u_t)$, $t = 0, 1 \ldots$, where X_t, X_{t+1} are fuzzy states at control stages t and $t+1$, respectively, defined in $X = \{s_1, \ldots, s_n\}$, i.e. $X_t, X_{t+1} \in \mathcal{X}$, and $u_t \in U = \{c_1, \ldots, c_m\}$ is a nonfuzzy control at control stage t, Notice that we assume here a fuzzy system under control but a nonfuzzy control; this is done for simplicity since the controls will correspond to branches (edges) in a decision tree. We should, however, bear in mind that one can also assume fuzzy controls, $U_t \in \mathcal{U}$, and then use some (finite, possibly low) number of predefined reference fuzzy controls. This will not be considered here but the method presented works analogously in that case.

At each t, the control $u_t \in U$ is subjected to a fuzzy constraint $\mu_{C^t}(u_t)$, and on the final fuzzy state attained a fuzzy goal $\mu_{G^N}(x_N)$ is imposed. The initial fuzzy state is $X_0 \in \mathcal{X}$. The final fuzzy state $X_N \in \mathcal{X}$ evidently cannot be introduced directly into the fuzzy goal $\mu_{G^N}(x_N)$, and hence a "trickery" outlined in the beginning of this section is employed.

The problem is to find an optimal sequence of (nonfuzzy) controls $u_0^*, \ldots, u_{N-1}^*$ such that (see (30))

$$\mu_D(u_0^*, \ldots, u_{N-1}^* \mid X_0) = \\ = \max_{u_0, \ldots, u_{N-1}} [\mu_{C^0}(u_0) \wedge \ldots \wedge \mu_{C^{N-1}}(u_{N-1}) \wedge \mu_{\overline{G}^N}(X_N)] \tag{36}$$

It is clear that this problem satisfies all the conditions of type (17)–(19) from Section 3.1.2. which form a basis of the branch-and-bound procedure. The algorithm is basically the same, with obvious replacements.

We have assumed here that the control is nonfuzzy. This has made it possible to directly construct the corresponding decision tree. In the case of fuzzy controls, one has to assume some (finite, possibly low) number of *reference fuzzy controls*. All the controls are then approximated by these reference fuzzy controls, and the branches of the decision tree are associated with the particular reference fuzzy control. This manipulation makes it possible to use the branch-and-bound algorithm presented above. However, we should bear in mind that we obtain here optimal reference fuzzy controls, which are not the same as the "real" fuzzy controls. So, to implement these reference fuzzy controls obtained we need to employ some procedure to infer proper "real" fuzzy controls. Kacprzyk's (1993a–c) interpolative reasoning based scheme presented below may be used here.

3.2.3. Solution by interpolative reasoning

Basically, Kacprzyk's (1993a–c) interpolative reasoning scheme is applied rather to the dynamic programming approach presented in Section 3.2.1., but can also be employed for the branch-and-bound approach presented in Section 3.2.2..

In Kacprzyk's (1993a–c) approach, a very small number of "non-overlapping" reference fuzzy states and controls is assumed, and in their terms an auxiliary (much simpler!) control problem is formulated. Its solution yields an auxiliary optimal control policy relating optimal reference fuzzy controls to reference fuzzy states. Such a policy is equated with a fuzzy relation which is then used to determine an auxiliary optimal control (not necessarily the reference one) for a particular fuzzy state (not necessarily the reference one).

Then, the (auxiliary) optimal solution (control) obtained in some way is adjusted to become a "real" optimal fuzzy control, by using some interpolation.

For a detailed description of this technique we refer the reader to Kacprzyk's (1997) book. In general, the method works well though is somehow complicated and difficult to implement.

In the next section we will present a conceptually and numerically simple genetic algorithm for solving the problem considered.

4. A genetic algorithm for the solution of the multistage fuzzy control problem with a deterministic system under control

From the previous sections we have learned that the solution of the multistage control problem considered (12) may be really difficult for practical problems of a non-trivial size, in spite of being relatively simple conceptually. Though this has been particularly true for dynamic programming, plagued by its inherent "curse of dimensionality", the same can also be said of branch-and-bound. On the other hand, Francelin and Gomide's (1992, 1993) and Francelin, Gomide and Kacprzyk's (1995) neural network approach is conceptually somehow complicated.

In the recent Kacprzyk's (1995a–c) papers the use of a genetic algorithm was proposed. This has provided, first, a conceptually simple and general solution tool, and, second, it has turned out to be numerically efficient. The essence of that approach, and its further extension, will be presented below. First, we will outline the basic idea of the genetic algorithm to be employed, then show its application to the solution of the problem considered, and finally present computational results.

4.1. Idea of a genetic algorithm

Genetic algorithms are stochastic algorithms whose search methods "mirror" some phenomena underlying natural evolution processes, notably *genetic inheritance* and the Darwinian *survival of the fittest.*

In our context, by an *individual* we will mean a particular solution, i.e. particular values of controls at the consecutive control stages, $u_0, \ldots, u_{N-1}$. It is *evaluated* by the fuzzy decision (11), which is here the so-called *fitness function.*

A set of potential solutions will be termed a *population*, and its size will be assumed fixed. So, we initially assume some (e.g., randomly generated) potential solutions (the initial population). Then, some members of the population, who play the role of parents, will undergo *reproduction* through the so-called *crossover* and *mutation* to produce their off-springs (children), i.e. some new solutions. Then, the best ones (the fittest) will "survive", i.e. will be used while repeating this process. Finally, one may expect to find a very good (if not optimal) solution.

The structure of a genetic algorithm may be portrayed as follows:

begin
$t =: 0$

set the initial population $P(t)$
evaluate strings in $P(t)$
while termination condition is not fulfilled do:
begin
$t := t + 1$
select current population $P(t)$ from $P(t-1)$
perform reproduction on elements of $P(t)$
calculate the evaluation function for each element of $P(t)$
end
end

Its basic elements:

- how to represent a potential solution,
- how to create (generate) an initial population,
- how to define the fitness (evaluation) function,
- how to perform the reproduction (crossover and mutation), and
- how to choose some parameters,

will now be clarified on a simple example.

First, the potential solutions are here sequences of controls such as u_0, u_1, u_2 for a control process with the termination time $N = 3$. If now $u_0, u_1, u_2 \in \{0, 1, \ldots, 7\}$, then there may be the following solution candidates:

Solution 1: $(2, 4, 5)$
Solution 2: $(1, 7, 6)$
Solution 3: $(3, 2, 5)$

which are represented in binary notation (i.e. as binary strings) as, respectively:

Solution 1:	0	1	0	1	0	0	1	0	1
Solution 2:	1	0	0	1	1	1	0	1	1
Solution 3:	1	1	0	0	1	0	1	0	1

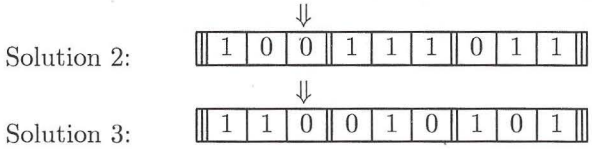
Each of the above three solution candidates (generated, e.g., randomly) is then evaluated by using the fuzzy decision (11) (or (10)). For instance, we obtain

$$e_1 = \mu_D(2, 4, 5 \mid .) = 0.7$$
$$e_2 = \mu_D(1, 7, 6 \mid .) = 0.3$$
$$e_3 = \mu_D(3, 2, 5 \mid .) = 0.9$$

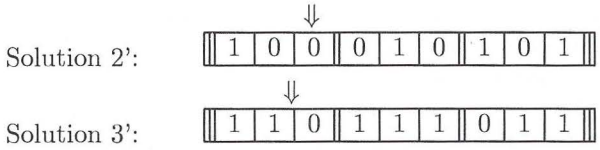
Now, the probability of selection of the solution candidate $i \in \{1, 2, 3\}$ is defined as

$$p_i = \frac{e_i}{e_1 + e_2 + e_3} \tag{37}$$

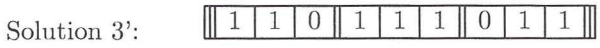
Then, suppose that based on this selection probability, p_i, the solutions 2 and 3 are selected out of the initial population to be the "parents" who are next subjected to the two basic operations to "produce" their offsprings (children). The first operation is *crossover*. We generate, using a *crossover probability*, some point (bit number) in the binary string from 1 to the length of the string (9 in our case), e.g., 3 as shown by a vertical arrow as



The crossover exchanges the respective bits between the two parents from bit 4 on which yields the following two new candidate solutions:



One of the two above candidates is selected at random and taken as the new individual, for instance Solution 3', i.e.



Next, using a prespecified *mutation probability*, we select a point in the above binary string, say 5, and change this bit in the string to the opposite value (i.e. 0 to 1 and vice versa) and this new candidate solution is introduced into the new population.

The process of random selection of the parents, crossover, and mutation is repeated until the new population of the predefined size is obtained. Then, the whole process is repeated, and is continued until some termination condition is satisfied as, e.g., the maximum number of iterations or a time limit.

The above process is conceptually and implementationally simple, and usually leads to good results. Due to random mechanisms widely employed, it escapes from local optima, and helps find a globally best (optimal) solution.

The idea of a genetic algorithm presented above is basic, and many modifications of both the crossover and mutation, as well as new operations, have been proposed, see Davis (1991) or Michalewicz (1994). Moreover, let us remark that though the binary representation of solutions (binary coding) is traditionally employed, one can well use a real coding of solutions in which solution candidates are represented as strings of real numbers (the subsequent controls). Such a real coding will be employed in this paper with crossover and mutation being direct derivatives of the traditional ones.

4.2. Using the genetic algorithm for solving the multistage fuzzy control problem with a detrministic system under control

To present the idea of Kacprzyk's (1995a–c) genetic-algorithm-based approach, it may be expedient to briefly restate the problem considered.

First, the deterministic system under control is given by the state transition equation (9), i.e.

$$x_{t+1} = f(x_t, u_t), \qquad t = 0, 1, \ldots \tag{38}$$

where: $x_t, x_{t+1} \in X = \{s_1, \ldots, s_n\}$ is the *state* (output) at control stages t and $t+1$, respectively, and $u_t \in U = \{c_1, \ldots, c_m\}$ is the *control* (input) at t. The initial state is $x_0 \in X$, and the (finite) termination time N is fixed and specified in advance.

At each control stage t, the control $u_t \in U$ is subjected to a *fuzzy constraint* $\mu_{C^t}(u_t)$, and on the final state $x_N \in X$ a *fuzzy goal* $\mu_{G^N}(x_N)$ is imposed [fuzzy goals at the subsequent t's may also be assumed, and the reasoning remains valid – cf. (10)].

The fitness (evaluation or performance) function is the fuzzy decision (11), i.e.

$$\begin{aligned} &\mu_D(u_0, \ldots, u_{N-1} \mid x_0) = \\ &\quad = \mu_{C^0}(u_0) \wedge \ldots \wedge \mu_{C^{N-1}}(u_{N-1}) \wedge \mu_{G^N}(x_N) \end{aligned} \tag{39}$$

and the problem is [cf. (12)] to find an *optimal sequence of controls*, $u_0^*, \ldots, u_{N-1}^*$, such that

$$\begin{aligned} &\mu_D(u_0^*, \ldots, u_{N-1}^* \mid x_0) = \\ &\quad = \max_{u_0, \ldots, u_{N-1}} \mu_D(u_0, \ldots, u_{N-1} \mid x_0) = \max_{u_0, \ldots, u_{N-1}} [\mu_{C^0}(u_0) \wedge \ldots \\ &\qquad \ldots \wedge \mu_{C^{N-1}}(u_{N-1}) \wedge \mu_{G^N}(x_N)] \end{aligned} \tag{40}$$

where $x_N = f(x_{N-1}, u_{N-1})$ by the state transition equation (38), and "$\wedge$" (minimum) may be replaced by another operation, notably a t-norm, Kacprzyk (1997).

The basic elements of the genetic algorithm to be used for solving the above problem are meant as follows:

- the problem is represented by strings of controls $u_0, \ldots, u_{N-1}$, and we use real coding;
- the fitness function is the fuzzy decision (39),
- standard random selections of elements from the consecutive populations, standard concepts of crossover and mutation (applied to real coded strings), and a standard termination condition, mainly a predefined number of iterations, or iteration-to-iteration improvement lower than a threshold is used;

Further, we assume that:

- controls are "evenly spaced" real numbers in $[0, 1]$ corresponding to $c_1, \ldots, c_m$, and
- states are defined as "evenly spaced" real numbers from $[0, 1]$ corresponding to $s_1, \ldots, s_n$.

The genetic algorithm works now as follows:

begin
$t =: 0$
set the initial population $P(t)$ which consists of
randomly generated strings of controls
(i.e. of randomly generated real numbers from $[0, 1]$);
for each $u_0, \ldots, u_{N-1}$ in each
string in the population $P(t)$,
find the resulting x_{t+1}
by using the state transition
equation $x_{t+1} = f(x_t, u_t)$,
and use the evaluation function (39)
$\mu_D(u_0, \ldots, u_{N-1} \mid x_0$ to evaluate each string in $P(t)$;
while $t <$ *maximum number of iterations* **do**
begin
$t := t + 1$
assign the probabilities to each string in $P(t-1)$
which are proportional to the value of the evaluation
function for each string;
randomly (using those probabilities) generate
the new population $P(t)$;
perform crossover and mutation on the strings in $P(t)$;
calculate the value of the evaluation function (39) for each string in $P(t)$.
end
end

We will illustrate now this algorithm by a simple example.

Example 4 Suppose that $X = \{s_1, \ldots, s_{20}\}$, $U = \{c_1, \ldots, c_{32}\}$, $N = 10$, and $x_0 = s_1$.

The state transition equation (38) is given as

$x_{t+1} = f(x_t, u_t) =$

	$x_t = s_1$	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	...
$u_t = c_1$	s_1	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	...
c_2	s_2	s_3	s_3	s_3	s_6	s_7	s_8	s_9	s_{10}	s_{11}	...
c_3	s_2	s_3	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	...
...	...	...	...	...	...	...	...	...	...	...	...
c_{30}	s_4	s_5	s_6	s_7	s_{10}	s_{11}	s_{12}	s_{14}	s_{15}	s_{15}	...
c_{31}	s_5	s_6	s_7	s_8	s_{10}	s_{11}	s_{12}	s_{14}	s_{15}	s_{16}	...
c_{32}	s_6	s_7	s_8	s_{10}	s_{11}	s_{12}	s_{14}	s_{15}	s_{16}	s_{17}	...

...

	...	$x_t = s_{11}$	s_{12}	s_{13}	s_{14}	s_{15}	s_{16}	s_{17}	s_{18}	s_{19}	s_{20}
$u_t = c_1$	...	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}	s_{16}	s_{17}	s_{18}	s_{19}
c_2	...	s_{12}	s_{13}	s_{14}	s_{15}	s_{16}	s_{17}	s_{18}	s_{19}	s_{20}	s_{20}
c_3	...	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}	s_{16}	s_{17}	s_{18}	s_{19}	s_{20}
...	...	...	...	...	...	...	...	...	...	...	
c_{30}	...	s_{16}	s_{17}	s_{19}	s_{19}	s_{20}	s_{20}	s_{20}	s_{20}	s_{20}	s_{20}
c_{31}	...	s_{16}	s_{17}	s_{18}	s_{19}	s_{20}	s_{20}	s_{20}	s_{20}	s_{20}	s_{20}
c_{32}	...	s_{17}	s_{18}	s_{19}	s_{20}	s_{20}	s_{20}	s_{20}	s_{20}	s_{20}	s_{20}

The fuzzy constraints and fuzzy goals at the consecutive control stages are given as trapezoid fuzzy numbers in $[0, 1]$, i.e. are equated with the 4-tuples.

The fuzzy constraints and goals are therefore assumed to be:

$$
\begin{aligned}
C^0 &= (c_1, c_1, c_4, c_{32}) & G^1 &= (s_1, s_2, s_7, s_9)\\
C^1 &= (c_1, c_1, c_7, c_{32}) & G^2 &= (s_2, s_3, s_9, s_{11})\\
C^2 &= (c_1, c_1, c_9, c_{32}) & G^3 &= (s_3, s_5, s_9, s_{11})\\
C^3 &= (c_1, c_1, c_{10}, c_{31}) & G^4 &= (s_4, s_7, s_{12}, s_{14})\\
C^4 &= (c_1, c_1, c_{12}, c_{32}) & G^5 &= (s_5, s_8, s_{14}, s_{16})\\
C^5 &= (c_1, c_1, c_{13}, c_{32}) & G^6 &= (s_6, s_{10}, s_{16}, s_{18}\\
C^6 &= (c_1, c_1, c_{15}, c_{32}) & G^7 &= (s_7, s_{11}s_{16}, s_{18})\\
C^7 &= (c_1, c_1, c_{17}, c_{32}) & G^8 &= (s_9, s_{14}, s_{18}, s_{20})\\
C^8 &= (c_1, c_1, c_{18}, c_{32}) & G^9 &= (s_{11}, s_{16}, s_{20}, s_{20})\\
C^9 &= (c_1, c_1, c_{20}, c_{32}) & G^{10} &= (s_{14}, s_{18}, s_{20}, s_{20})
\end{aligned}
$$

The main parameters are assumed to be:

- the population size is 250,
- the number of trials is 32,000,
- the crossover rate is 0.6, and
- the mutation rate is 0.001.

We obtain 3 best (optimal) results (starting from $x_0 = s_1$):

$$
\begin{aligned}
&u_0^* = c_3 \quad u_1^* = c_4 \quad u_2^* = c_4 \quad u_3^* = c_5 \quad u_4^* = c_8\\
&u_5^* = c_{11} \quad u_6^* = c_{12} \quad u_7^* = c_{14} \quad u_8^* = c_{15} \quad u_9^* = c_{15}
\end{aligned}
$$

$$
\begin{aligned}
&u_0^* = c_2 \quad u_1^* = c_5 \quad u_2^* = c_3 \quad u_3^* = c_5 \quad u_4^* = c_8\\
&u_5^* = c_{11} \quad u_6^* = c_{13} \quad u_7^* = c_{14} \quad u_8^* = c_{14} \quad u_9^* = c_{15}
\end{aligned}
$$

$$
\begin{aligned}
&u_0^* = c_3 \quad u_1^* = c_6 \quad u_2^* = c_3 \quad u_3^* = c_5 \quad u_4^* = c_8\\
&u_5^* = c_{11} \quad u_6^* = c_{12} \quad u_7^* = c_{14} \quad u_8^* = c_{15} \quad u_9^* = c_{15}
\end{aligned}
$$

for which the corresponding value of the fuzzy decision (39) is

$$\mu_D(u_0^*, \ldots, u_9^* \mid s_1) = 1$$

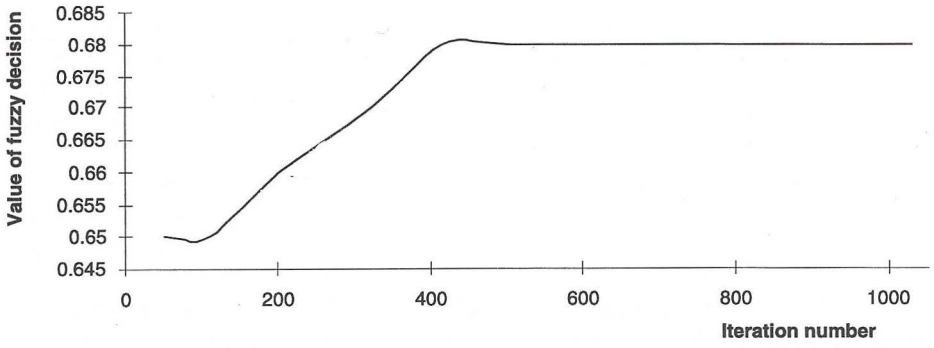


Figure 3. The value of the fuzzy decision obtained in the course of iterations in Example 4

The next best result is

$$u_0^* = c_3 \quad u_1^* = c_6 \quad u_2^* = c_3 \quad u_3^* = c_5 \quad u_4^* = c_8$$
$$u_5^* = c_{12} \quad u_6^* = c_{13} \quad u_7^* = c_{15} \quad u_8^* = c_{12} \quad u_9^* = c_{14}$$

and its corresponding value of the fuzzy decision (39) is

$$\mu_D(u_0^*, \ldots, u_0^* \mid s_1) = 0.973684$$

while the tenth best result is

$$u_0^* = c_2 \quad u_1^* = c_4 \quad u_2^* = c_3 \quad u_3^* = c_5 \quad u_4^* = c_8$$
$$u_5^* = c_{12} \quad u_6^* = c_{13} \quad u_7^* = c_{15} \quad u_8^* = c_{12} \quad u_9^* = c_{14}$$

and its corresponding value of the fuzzy decision (39) is

$$\mu_D(u_0^*, \ldots, u_9^* \mid s_1) = 0.973684$$

In Figure 3 the best values of the fuzzy decision (39) obtained in the course of iterations are shown, and it may readily be seen that the optimal solution has been attained quite early, so that 32,000 iterations assumed have not been necessary. □

Note that the results obtained are clearly very good, with the best results obtained being optimal indeed. Obviously, the solution of the problem considered, i.e. with the assumed number of control stages, and the size of the state and control spaces, by using any of the other techniques (dynamic programming, branch-and-bound and neural networks) would certainly be more complicated from the conceptual, implementational and computational points of view.

5. A genetic algorithm for the solution of the multistage fuzzy control problem with a fuzzy system under control

As it can easily be seen from Section 3.2., the case of multistage fuzzy control with a fuzzy system under control is more difficult – both conceptually and numerically – than the case of a deterministic system. Though the basic solution techniques proposed, i.e. Baldwin and Pilsworth's (1982) dynamic programming, possibly enhanced with Kacprzyk's (1993a–c) interpolative reasoning, and Kacprzyk's (1979) branch-and-bound, do share the same numerical difficulties with the case of the deterministic system, these are clearly more pronounced when a fuzzy system is assumed. This does clearly suggest that the use of a genetic algorithm can here be even more justified as it was proposed by Kacprzyk (1995a–c).

We employ here the same basic framework of a genetic algorithm as presented earlier in Section 4.1. By an *individual* we mean a particular solution, i.e. the particular values of the fuzzy controls at the consecutive control stages, $U_0, \ldots, U_{N-1}$. An individual is evaluated by the fuzzy decision (29), which is here the *fitness function*. A set of potential solutions is termed a *population* which is assumed to be of a fixed size. The population is generated and transformed analogously as outlined in Section 4.1.

The general structure of a genetic algorithm is analogous as presented in Section 4.1., and the basic problems are also the same, i.e.

- how to represent a potential solution,
- how to create (generate) an initial population,
- how to define a fitness function,
- how to perform the reproduction (crossover and mutation), and
- how to choose some parameters.

Before the description of the genetic algorithm tailored to the problem considered, we will briefly restate the problem formulation (cf. Section 3.2.).

The dynamics of a fuzzy system under control is given by a state transition equation (21), i.e.

$$X_{t+1} = F(X_t, U_t), \qquad t = 0, 1, \ldots \tag{41}$$

where $X_t, X_{t+1} \in \mathcal{X}$ are fuzzy states at control stage t and $t+1$, and $U_t \in \mathcal{X}$ is a fuzzy control at t, $t = 0, 1, \ldots$; (41) is equivalent to a conditioned fuzzy set $\mu_{X_{t+1}}(x_{t+1} \mid x_t, u_t)$ or a fuzzy relation in $X \times X \times U$ (see Kacprzyk, 1997).

At each t, the fuzzy control applied U_t is subjected to a fuzzy constraint $\mu_{C^t}(u_t)$, and on the resulting fuzzy state X_{t+1} a fuzzy goal $\mu_{G^{t+1}}(x_{t+1})$ is imposed, $t = 0, 1, \ldots, N-1$.

Both the fuzzy controls, U_t's, and fuzzy states, X_{t+1}'s, are now fuzzy, hence (see Section 3.2.) their grades of membership in the fuzzy constraints and goals cannot be directly determined as the values of $\mu_{C^t}(u_t)$ and $\mu_{G^{t+1}}(x_{t+1})$. In the

source papers (Kacprzyk's, 1995a–b), on which our discussion is based, it was generally used as a measure of dissimilarity (23), i.e.

$$\mu_{\overline{C}^t}(U_t) = 1 - diss(C^t, U_t), t = 0, 1, \ldots, N-1 \tag{42}$$

and, to be more specific, Kaufmann and Gupta's (1985) dissimilarity index (46), to be shown below.

Thus, the fuzzy decision is

$$\begin{aligned}&\mu_D(U_0, \ldots, U_{N-1} \mid X_0) = \\ &\quad = \mu_{\overline{C}^0}(U_0) \wedge \mu_{\overline{G}^1}(X_1) \wedge \ldots \wedge \mu_{\overline{C}^{N-1}}(U_{N-1}) \wedge \mu_{\overline{G}^N}(X_N)\end{aligned} \tag{43}$$

and the problem is to find $U_0^*, \ldots, U_{N-1}^*$ such that

$$\mu_D(U_0^*, \ldots, U_{N-1}^* \mid X_0) = \max_{U_0, \ldots, U_{N-1}} \mu_D(U_0, \ldots, U_{N-1} \mid X_0) \tag{44}$$

Due to the specifics of the problem with a fuzzy system, the basic elements of the genetic algorithm are meant as:

- the problem is represented by strings of fuzzy controls $U_0, \ldots, U_{N-1}$ (real coding), and we use triangular fuzzy numbers to represent fuzzy controls (moreover, some reference fuzzy controls, $\overline{U}_0, \ldots, \overline{U}_{N-1}$ are also used);
- the fitness (evaluation) function is (43), i.e.

$$\begin{aligned}&\mu_D(U_0, \ldots, U_{N-1} \mid X_0) = \mu_{\overline{C}^0}(U_0) \wedge \mu_{G^1}(X_1) \wedge \ldots \\ &\quad \ldots \wedge \mu_{\overline{C}^{N-1}}(U_{N-1}) \wedge \mu_{\overline{G}^N}(X_N)\end{aligned} \tag{45}$$

  and for its calculation [of $\mu_{\overline{C}^t}(U_t)$ and $\mu_{\overline{G}^{t+1}}(X_{t+1})$] we use the degree of dissemblance by Kaufmann and Gupta (1985) which is defined, for triangular fuzzy numbers, as: if A and B are triangular fuzzy numbers, then the *degree of dissemblance* of A and B is

$$diss(A, B) = \int_{\alpha=0}^{1} \frac{1}{2}(\mid \underline{a}^\alpha - \underline{b}^\alpha \mid + \mid \overline{a}^\alpha - \overline{b}^\alpha \mid) \, d\alpha \tag{46}$$

  where $[\underline{a}^\alpha, \overline{a}^\alpha]$ and $[\underline{b}^\alpha, \overline{b}^\alpha]$ are the so-called α-cuts (intervals) of A and B, $\forall \alpha \in (0, 1]$; the α-cut of a fuzzy set A in $X = \{x\}$, A_α, is defined as the nonfuzzy set $A_\alpha = \{x \in X : \mu_A(x) \geq \alpha\}$, $\forall \alpha \in (0, 1]$.

  Therefore, if

  $$f_t(U_t, C^t, X_{t+1}, G^{t+1}) = [1 - diss(U_t, C^t)] \wedge [1 - diss(X_{t+1}, G^{t+1})],$$

  $t = 0, 1, \ldots N-1$, then the fitness function (45) becomes

$$\begin{aligned}&f(U_0, X_1, \ldots, U_{N-1}, X_N) = \\ &\quad = \mu_D(U_0, \ldots, U_{N-1} \mid X_0) = f_0(U_0, C^0, X_1, G^1) \wedge \ldots \\ &\quad \ldots \wedge f_{N-1}(U_{N-1}, C^{N-1}, X_N, G^N)\end{aligned} \tag{47}$$

- standard random selections of elements from the consecutive populations, standard crossover and mutation (evidently, applied to real coded strings), and a standard termination condition, mainly a predefined number of iterations, or iteration-to-iteration improvement lower than a threshhold) are used.

Further, we assume that:

- fuzzy controls are fuzzy sets in $[0,1]$ defined as triangular fuzzy numbers in $[0,1]$, i.e. as the triples (a,b,c), $0 \leq a \leq b \leq x \leq 1$; the left and right spreads (widths) are assumed to be equal to 5% each, for simplicity, hence only the mean value (b) is generated; moreover, 10 reference fuzzy controls are introduced;
- fuzzy states are defined as fuzzy sets in $X = \{s_1, \ldots, s_{10}\}$;
- fuzzy constraints are defined as trapezoid fuzzy numbers in $[0,1]$;
- fuzzy goals are defined as fuzzy sets in $\{s_1, \ldots, s_{10}\}$;
- the dynamics of the fuzzy system under control (41), i.e. the state transition equation, is defined as a set of fuzzy relations $R_{\overline{U}}$ in $S \times S$, for each of the reference fuzzy control (we need reference fuzzy controls as otherwise we would need infinitely many fuzzy relations, for each possible fuzzy control); so, to choose an appropriate table to determine the state transition, first we find a reference fuzzy control that is the closest [in the sense of the dissemblance index used (46)] to the current control, and then we take its corresponding fuzzy relation to find the resulting fuzzy state X_{t+1}.

The genetic algorithm employed is as follows:

begin
  $t := 0$
  set the initial population $P(t)$
    which consists of randomly generated
    strings of triangular fuzzy controls
    (i.e. of randomly generated mean
    values from $[0,1]$, with 5% left and right spreads);
  for each $U_0, \ldots, U_{N-1}$ in each string in the population $P(t)$:
    find the resulting X_{t+1} (by finding first the closest reference
      fuzzy control to choose an appropriate relation
      which is followed by using
      the compositional rule of inference),
    and use the evaluation function (47)
      to evaluate each string in $P(t)$;
  while $t <$ *maximum number of iterations* do
  begin
    $t := t + 1$
    assign the probabilities to each
      string in $P(t-1)$ which are proportional
      to the value of the evaluation function for each string;
    randomly (using those probabilities)
      generate the new population $P(t)$;
    perform crossover and mutation on the strings in $P(t)$;
    calculate the evaluation function (47) for each string in $P(t)$.

end
end

To illustrate this algorithm we will now solve a simple example.

Example 5 Suppose that: $N = 10$, $X = \{s_1, \ldots, s_{10}\}$, the controls are triangular fuzzy numbers in $[0, 1]$, and there are 10 "equally-spaced" (with the mean values at 0.1, ..., 0.9, 1) reference fuzzy controls defined as the trapezoid fuzzy numbers in $[0, 1]$ as follows:

$$\begin{array}{ll} \overline{C}_1 = (0.0, 0.1, 0.1, 0.2) & \overline{C}_2 = (0.1, 0.2, 0.2, 0.3) \\ \overline{C}_3 = (0.2, 0.3, 0.3, 0.4) & \overline{C}_4 = (0.3, 0.4, 0.4, 0.5) \\ \overline{C}_5 = (0.4, 0.5, 0.5, 0.6) & \overline{C}_6 = (0.5, 0.6, 0.6, 0.7) \\ \overline{C}_7 = (0.6, 0.7, 0.7, 0.8) & \overline{C}_8 = (0.7, 0.8, 0.8, 0.9) \\ \overline{C}_9 = (0.8, 0.9, 0.9, 1.0) & \overline{C}_{10} = (0.9, 1.0, 1.0, 1.0) \end{array}$$

The initial fuzzy state is $X_0 = 1.0/s_1 + 0.7/s_2 + 0.4/s_3 + 0.1/s_4$.

The fuzzy constraints at the particular control stages are also given as the following trapezoid fuzzy numbers:

$$\begin{array}{ll} \overline{C}^0 = (0.0, 0.0, 0.5, 0.8) & \overline{C}^1 = (0.0, 0.0, 0.5, 0.8) \\ \overline{C}^2 = (0.0, 0.0, 0.5, 0.8) & \overline{C}^3 = (0.0, 0.0, 0.5, 0.8) \\ \overline{C}^4 = (0.0, 0.0, 0.5, 0.8) & \overline{C}^5 = (0.0, 0.0, 0.5, 0.8) \\ \overline{C}^6 = (0.0, 0.0, 0.5, 0.8) & \overline{C}^7 = (0.0, 0.0, 0.5, 0.8) \\ \overline{C}^8 = (0.0, 0.0, 0.5, 0.8) & \overline{C}^9 = (0.0, 0.0, 0.5, 0.8) \end{array}$$

The fuzzy goals at the particular control stages are:

$$\begin{aligned} \overline{G}^1 &= 0.1/s_1 + 0.2/s_2 + 0.3/s_3 + 0.6/s_4 + \\ &\quad + 1.0/s_5 + 0.6/s_6 + 0.3/s_7 + 0.2/s_8 + 0.1/s_9 + 0.0/s_{10} \\ \overline{G}^2 &= 0.1/s_1 + 0.2/s_2 + 0.3/s_3 + 0.6/s_4 + \\ &\quad + 1.0/s_5 + 0.6/s_6 + 0.3/s_7 + 0.2/s_8 + 0.1/s_9 + 0.0/s_{10} \\ \overline{G}^3 &= 0.1/s_1 + 0.2/s_2 + 0.3/s_3 + 0.6/s_4 + \\ &\quad + 1.0/s_5 + 0.6/s_6 + 0.3/s_7 + 0.2/s_8 + 0.1/s_9 + 0.0/s_{10} \\ \overline{G}^4 &= 0.1/s_1 + 0.2/s_2 + 0.3/s_3 + 0.6/s_4 + 1.0/s_5 + 0.6/s_6 + \\ &\quad + 0.3/s_7 + 0.2/s_8 + 0.1/s_9 + 0.0/s_{10} \\ \overline{G}^5 &= 0.1/s_1 + 0.2/s_2 + 0.3/s_3 + 0.6/s_4 + \\ &\quad + 1.0/s_5 + 0.6/s_6 + 0.3/s_7 + 0.2/s_8 + 0.1/s_9 + 0.0/s_{10} \\ \overline{G}^6 &= 0.1/s_1 + 0.2/s_2 + 0.3/s_3 + 0.6/s_4 + \\ &\quad + 1.0/s_5 + 0.6/s_6 + 0.3/s_7 + 0.2/s_8 + 0.1/s_9 + 0.0/s_{10} \end{aligned}$$

$$\overline{G}^7 = 0.1/s_1 + 0.2/s_2 + 0.3/s_3 + 0.6/s_4 + \\ + 1.0/s_5 + 0.6/s_6 + 0.3/s_7 + 0.2/s_8 + 0.1/s_9 + 0.0/s_{10}$$
$$\overline{G}^8 = 0.1/s_1 + 0.2/s_2 + 0.3/s_3 + 0.6/s_4 + \\ + 1.0/s_5 + 0.6/s_6 + 0.3/s_7 + 0.2/s_8 + 0.1/s_9 + 0.0/s_{10}$$
$$\overline{G}^9 = 0.1/s_1 + 0.2/s_2 + 0.3/s_3 + 0.6/s_4 + \\ + 1.0/s_5 + 0.6/s_6 + 0.3/s_7 + 0.2/s_8 + 0.1/s_9 + 0.0/s_{10}$$
$$\overline{G}^{10} = 0.1/s_1 + 0.2/s_2 + 0.3/s_3 + 0.6/s_4 + \\ + 1.0/s_5 + 0.6/s_6 + 0.3/s_7 + 0.2/s_8 + 0.1/s_9 + 0.0/s_{10}$$

The fuzzy state transitions (41), are specified as conditioned fuzzy sets for each particular reference fuzzy control, $\overline{C}_1, \ldots, \overline{C}_{10}$. Due to lack of space we will only present below the state transtion equations for the first and last reference fuzzy control, i.e. $\overline{C}_1$ and $\overline{C}_{10}$, and these are:

- for $\overline{C}_1$

	$x_{t+1} = s_1$	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
$x_t = s_1$	0.0	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.1
s_2	0.0	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.1
s_3	0.0	0.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.1
s_4	0.0	0.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.1
s_5	0.0	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.1
s_6	0.0	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.1
s_7	0.0	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.1
s_8	0.0	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.1
s_9	0.0	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.1
s_{10}	0.0	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.1

- ...
- for $\overline{C}_{10}$

	$x_{t+1} = s_1$	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
$x_t = s_1$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
s_2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
s_3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
s_4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
s_5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
s_6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
s_7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
s_8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
s_9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
s_{10}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

The main parameters are:

- the population size is 50,
- the maximum number of iterations (termination condition) is 1000,
- the crossover rate is 0.6, and
- the mutation rate is 0.001.

The ten best results obtained may be summarized as follows:

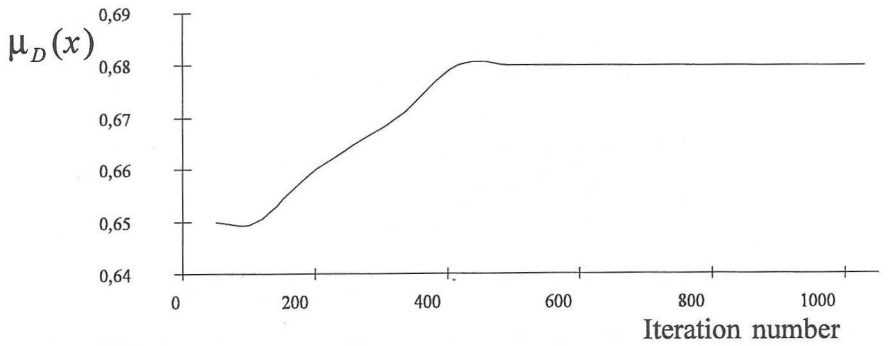


Figure 4. The value of the fuzzy decision obtained in the course of iterations in Example 5

- the best fuzzy controls at $t = 0, 1, \ldots, 9$ are:
  $U_0^* = (0.4885, 0.5142, 0.5399)$ $U_1^* = (0.5031, 0.5296, 0.5561)$
  $U_2^* = (0.4236, 0.4459, 0.4682)$ $U_3^* = (0.4842, 0.5097, 0.5352)$
  $U_4^* = (0.4651, 0.4895, 0.5140)$ $U_5^* = (0.4916, 0.5175, 0.5434)$
  $U_6^* = (0.3218, 0.3387, 0.3556)$ $U_7^* = (0.5225, 0.5500, 0.5775)$
  $U_8^* = (0.3451, 0.3633, 0.3815)$ $U_9^* = (0.2615, 0.2752, 0.2890)$
  and the value of the fuzzy decision (47) is
  $\mu_D(U_0^*, \ldots, U_9^* \mid X_0) = 0.681881$
- the second best result is
  $U_0 = (0.4885, 0.5142, 0.5399)$ $U_1 = (0.5031, 0.5296, 0.5561)$
  $U_2 = (0.4236, 0.4459, 0.4682)$ $U_3 = (0.4842, 0.5097, 0.5352)$
  $U_4 = (0.4651, 0.4895, 0.5140)$ $U_5 = (0.4916, 0.5175, 0.5434)$
  $U_6 = (0.3218, 0.3387, 0.3556)$ $U_7 = (0.5225, 0.5500, 0.5775)$
  $U_8 = (0.3451, 0.3633, 0.3815)$ $U_9 = (0.2615, 0.2752, 0.2890)$
  and the value of the fuzzy decision (47) is
  $\mu_D(U_0^*, \ldots, U_9^* \mid X_0) = 0.681881$
- while the tenth best result is:
  $U_0^* = (0.2510, 0.2642, 0.2774)$ $U_1^* = (0.4758, 0.5008, 0.5259)$
  $U_2^* = (0.4855, 0.5111, 0.5366)$ $U_3^* = (0.5432, 0.5718, 0.6004)$
  $U_4^* = (0.4780, 0.5032, 0.5284)$ $U_5^* = (0.5182, 0.5455, 0.5728)$
  $U_6^* = (0.5100, 0.5368, 0.5637)$ $U_7^* = (0.3316, 0.3491, 0.3665)$
  $U_8^* = (0.4639, 0.4883, 0.5127)$ $U_9^* = (0.3816, 0.4016, 0.4217)$
  and the value of the fuzzy decision (47) is
  $\mu_D(U_0^*, \ldots, U_9^* \mid X_0) = 0.679795$

The best values of the fuzzy decision (47) obtained in the course of iterations are shown in Figure 4, and it may readily be see that the best (maybe optimal) solution has been attained quite early, before the 1,000 iterations assumed. □

In general, also for many different problems solved, the algorithm has proven to be efficient.

6. Concluding remarks

In this paper we have shown the use of a genetic algorithm for the solution of multistage fuzzy control problems with a fixed and specified termination time, and with a deterministic and fuzzy systems under control. The genetic algorithm proposed is conceptually simpler than the traditionally employed techniques which are mainly based on dynamic programming and branch-and-bound. Moreover, it is computationally efficient. It seems that genetic algorithms may provide a viable alternative for the multistage fuzzy control problems.

References

BALDWIN, J.F. and PILSWORTH, B.W. (1982) Dynamic programming for fuzzy systems with fuzzy environment. *Journal of Mathematical Analysis and Applications*, **85**, 1–23.

BELLMAN, R.E. and ZADEH, L.A. (1970) Decision making in a fuzzy environment. *Management Science*, **17**, 141–164.

BRITOV, G.S. and REZNIK, L.K. (1981) Optimal control of linear fuzzy systems (in Russian). *Automation and Remote Control*, **42**, 462–465.

CAO, S.G. and REES, N.W. (1995) Identification of fuzzy models. *Fuzzy Sets and Systems*, **74**, 307–320.

CHANG, R.L.P. and PAVLIDIS, T. (1977) Fuzzy decision tree algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-7**, 28–35.

CHANG, S.S.L. (1969A) Fuzzy dynamic programming and the decision making process. *Proceedings of the Third Princeton Conference on Information Sciences* (Princeton, NJ, USA).

CHANG, S.S.L. (1969B) Fuzzy dynamic programming and approximate optimization of partially known systems. *Proceedings of the Second Hawaii International Conference on Systems Science* (Honolulu, HI, USA).

CHANG, S.S.L. and ZADEH, L.A. (1972) On fuzzy mapping and control. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-2**, 30–34.

CHEN, Y.Y. and TSAO, T.C. (1989) A description of the dynamical behavior of fuzzy systems. *IEEE Transactions on Systems, Man and Cybernetics*, **19**, 745–755.

CZOGAŁA, E. and PEDRYCZ, W. (1981) On identification of fuzzy systems and its application in control problems. *Fuzzy Sets and Systems*, **6**, 73–83.

CZOGAŁA, E. and PEDRYCZ, W. (1984) Identification and control problems in fuzzy systems. *TIMS Studies in the Management Sciences*, **20**, 447–466.

DAVIS, L. (1991) *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York.

DELGADO, M., KACPRZYK, J., VERDEGAY, J.L. and VILA, M.A., eds. (1994)*Fuzzy Optimization: Recent Advances*, Physica-Verlag, Heidelberg.

DRIANKOV ,D., HELLENDOORN, H. and REINFRANK, M. (1993) *An Introduction to Fuzzy Control.* Springer-Verlag, Berlin.

ESOGBUE, A.O. (1991) Computational aspects and applications of a branch and bound algorithm for fuzzy multistage decision processes. *Computers and Mathematics with Applications*, **21**, 117–127.

ESOGBUE, A.O. and BELLMAN, R.E. (1984) Fuzzy dynamic programming and its extensions. *TIMS/Studies in the Management Sciences*, **20**, 147–167.

ESOGBUE, A.O., FEDRIZZI, M. and KACPRZYK, J. (1988) Fuzzy dynamic programming with stochastic systems. In J. Kacprzyk and M. Fedrizzi, eds., *Combining Fuzzy Imprecision with Probabilistic Uncertainty in Decision Making.* Springer-Verlag, Berlin/New York, 266–285.

FILEV, D. and ANGELOV, P. (1992) Fuzzy optimal control. *Fuzzy Sets and Systems*, **47**, 151–156.

FRANCELIN, R.A. and GOMIDE, F.A.C. (1992) Neural network to solve fuzzy discrete programming problems. Tech. Rep. 018/92, DCA/FEE UNICAMP, Campinas, Brazil.

FRANCELIN, R.A. and GOMIDE, F.A.C. (1993) A neural network for fuzzy decision making problems. *Proceedings of Second IEEE International Conference on Fuzzy Systems – FUZZ-IEEE'93* (San Francisco, CA, USA), **1**, 655–660.

FRANCELIN, R.A., GOMIDE, F.A.C. and KACPRZYK, J. (1995) A class of neural networks for dynamic programming. *Proceedings of Sixth International Fuzzy Systems Association World Congress (Saõ Paolo, Brazil)*, **II**, 221–224.

FUNG, L.W. and FU, K.S. (1977) Characterization of a class of fuzzy optimal control problems. In: M.M. Gupta, G.N. Saridis and B.R. Gaines, eds., *Fuzzy Automata and Decision Processes*, New York: North-Holland, 209–219.

KACPRZYK, J. (1977) Control of a nonfuzzy system in a fuzzy environment with a fuzzy termination time. *Systems Science*, **3**, 320–334.

KACPRZYK, J. (1978A) A branch-and-bound algorithm for the multistage control of a nonfuzzy system in a fuzzy environment. *Control and Cybernetics*, **7**, 51–64.

KACPRZYK, J. (1978B) Control of a stochastic system in a fuzzy environment with a fuzzy termination time. *Systems Science*, **4**, 291–300.

KACPRZYK, J. (1978C) Decision-making in a fuzzy environment with fuzzy termination time. *Fuzzy Sets and Systems*, **1**, 169–179.

KACPRZYK, J. (1979) A branch-and-bound algorithm for the multistage control of a fuzzy system in a fuzzy environment. *Kybernetes*, **8**, 139–147.

KACPRZYK, J. (1983A) *Multistage Decision Making under Fuzziness*, Verlag TÜV Rheinland, Cologne.

KACPRZYK, J. (1983B) A generalization of fuzzy multistage decision making and control via linguistic quantifiers. *International Journal of Control*, **38**, 1249–1270.

KACPRZYK, J. (1986) Towards 'human-consistent' multistage decision making and control models via fuzzy sets and fuzzy logic. Bellman Memorial Issue (A.O. Esogbue, Ed.), *Fuzzy Sets and Systems*, **18**, 299–314.

KACPRZYK, J. (1987) Stochastic systems in fuzzy environments: control. In: M.G. Singh, ed., *Systems and Control Encyclopedia*, Pergamon Press, Oxford, 4657–4661.

KACPRZYK, J. (1992) Fuzzy logic with linguistic quantifiers in decision making and control. *Archives of Control Sciences*, **1**, XXXVII, 127–141.

KACPRZYK, J. (1993A) Interpolative reasoning in optimal fuzzy control. *Proceedings of Second IEEE International Conference on Fuzzy Systems – FUZZ-IEEE'93* (San Francisco, CA, USA), **II**, 1259–1263.

KACPRZYK, J. (1993B) Fuzzy control with an explicit performance function using dynamic programming and interpolative reasoning. *Proceedings of First European Congress on Fuzzy and Intelligent Technologies – EUFIT'93* (Aachen, Germany), **3**, 1459–1463.

KACPRZYK, J. (1993C) Interpolative reasoning for computationally efficient optimal fuzzy control. *Proceedings of Fifth International Fuzzy Systems Association World Congress '93* (Seoul, Korea), **II**, 1270–1273.

KACPRZYK, J., (1994) Fuzzy dynamic programming – basic issues. In M. Delgado, J. Kacprzyk, J.-L. Verdegay and M.A. Vila, eds., *Fuzzy Optimization: Recent Advances*, Physica-Verlag, Heidelberg, 321–331.

KACPRZYK, J. (1995A) A genetic algorithm for the multistage control of a fuzzy system in a fuzzy environment. *Proceedings of Joint Third International IEEE Conference on Fuzzy Systems and Second International Symposium on Fuzzy Engineering – FUZZ-IEEE'95/IFES'95* (Yokohama, Japan), **III**, 1083–1088

KACPRZYK, J. (1995B) Multistage fuzzy control using a genetic algorithm. *Proceedings of Sixth World International Fuzzy Systems Association Congress* (Saõ Paolo, Brazil), **II**, 225–228

KACPRZYK, J. (1995C) A modified genetic algorithm for multistage control of a fuzzy system. *Proceedings of Third European Congress on Intelligent Techniques and Soft Computing – EUFIT'95* (Aachen, Germany), **1**, 463–466.

KACPRZYK, J. (1997) *Multistage Fuzzy Control.* Wiley, Chichester.

KACPRZYK, J. and ESOGBUE, A.O. (1996) Fuzzy dynamic programming: main developments and applications. *Fuzzy Sets and Systems*, **81**, 31–46.

KACPRZYK, J. and IWAŃSKI, C. (1987) A generalization of discounted multistage decision making and control through fuzzy linguistic quantifiers: an attempt to introduce commonsense knowledge. *International Journal of Control*, **45**, 1909–1930.

KACPRZYK, J., SAFTERUK, K. and STANIEWSKI, P. (1981) On the control of stochastic systems in a fuzzy environment over infinite horizon. *Systems Science*, **7**, 121–131.

KACPRZYK, J. and STANIEWSKI, P. (1980) A new approach to the control of stochastic systems in a fuzzy environment. *Archiwum Automatyki i Telemechaniki*, **XXV**, 433–443.

KACPRZYK, J. and STANIEWSKI, P. (1982) Long-term inventory policy-making through fuzzy decision-making models. *Fuzzy Sets and Systems*, **8**, 117–132.

KACPRZYK, J. and STANIEWSKI, P. (1983) Control of a deterministic system in a fuzzy environment over an infinite planning horizon. *Fuzzy Sets and Systems*, **10**, 291–298.

KACPRZYK, J. and STRASZAK, A. (1984) Determination of stable trajectories for integrated regional development using fuzzy decision models. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-14**, 310–313.

KACPRZYK, J. and YAGER, R.R. (1984) "Softer" optimization and control models via fuzzy linguistic quantifiers. *Information Sciences*, **34**, 157–178.

KAUFMANN, A. and GUPTA, M.M. (1985) *Introduction to Fuzzy Mathematics – Theory and Applications.* Van Nostrand Reinhold, New York.

KLIR, G.J. and YUAN, B. (1995) *Fuzzy Sets and Fuzzy Logic: Theory and Application.* Prentice-Hall, Englewood Cliffs, NJ.

KOMOLOV, S.V., MAKEEV, S.P., SEROV, G.P. and SHAKHNOV, I.F. (1979) On the problem of optimal control of a finite automaton with fuzzy constraints and fuzzy goal (in Russian). *Kybernetika* (Kiev), **6**, 30–34.

LIU, B.D. and ESOGBUE, A.O. (1996) Fuzzy criterion set and fuzzy criterion dynamic programming. *Journal of Mathematical Analysis and Applications*, **199**, 293–311.

MAMDANI, E.H. (1974) Application of fuzzy algorithms for the control of a simple dynamic plant. *Proceedings of IEE*, **121**, 1585–1588.

MICHALEWICZ, Z. (1994) *Genetic Algorithms + Data Structures = Genetic Programming.* Springer-Verlag, Heidelberg.

PEDRYCZ, W. (1993) *Fuzzy Control and Fuzzy Systems.* Research Studies Press/Wiley, Taunton/New York (Second edition).

PEDRYCZ, W., ed. (1996) *Fuzzy Modelling: Paradigms and Practice.* Kluwer, Boston.

ROCHA, A.F. (1993) *Neural Nets: A Theory for Brain and Machines.* Springer-Verlag, Heidelberg.

SUGENO, M. and YASUKAWA, T. (1993) A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems*, **FS-1**, 7–31.

YAGER, R.R. and FILEV, D.P. (1994) *Foundations of Fuzzy Control.* Wiley, New York.

YAGER, R.R. and KACPRZYK, J., eds. (1997) *The Ordered Weighted Averaging Operators: Theory, Methodology and Applications.* Kluwer, Boston.

ZADEH, L.A. (1965) Fuzzy sets. *Information and Control*, **8**, 338–353.

ZADEH, L.A. (1972) A rationale for fuzzy control. *Measurement and Control*, **34**, 3–4.

ZADEH, L.A. (1973) Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-2**, 28–44.

ZADEH, L.A. and KACPRZYK, J., eds. (1992) *Fuzzy Logic for the Management of Uncertainty*, Wiley, New York.

ZIMMERMANN, H.-J. (1976) Description and optimization of fuzzy systems. *International Journal of General Systems*, **2**, 209–215.

ZIMMERMANN, H.-J. (1996) *Fuzzy Set Theory and its Applications.* Kluwer, Boston (Third edition).