Raport Badawczy
Research Report

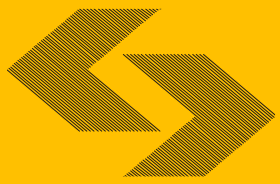
RB/72/2005

Optimal control of multistage deterministic, stochastic and fuzzy processes in the fuzzy environment via an evolutionary algorithm

J. Stańczak

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences



Control and Cybernetics

vol. **34** (2005) No. 2

Optimal control of multistage deterministic, stochastic and fuzzy processes in the fuzzy environment via an evolutionary algorithm

by

Jarosław Stańczak

Systems Research Institute
Polish Academy of Sciences
Newelska 6, 01-447 Warsaw, Poland
e-mail: stanczak@ibspan.waw.pl

Abstract: This paper deals with the problem of control of deterministic, stochastic and fuzzy systems with a fixed termination time and fuzzy constraints imposed on controls and states. Constrains imposed on the system are given as membership functions of particular fuzzy sets. Transition functions for controlled systems are given as a matrix of transitions between states for a deterministic object, a matrix of probabilities of transitions for a stochastic object and a matrix of membership functions of transitions for a fuzzy system.

An optimal (or sub-optimal) control is obtained using a specialized evolutionary algorithm (EA), which is a development over the previously used methods based on simple genetic algorithm. The specialized EA seems to be a very effective tool for solving such a class of optimization problems, comparing advantageously with the traditional simple genetic algorithm approach and with the previously used solutions like dynamic programming or branch and bound. The specialization of the applied EA is obtained using dedicated problem encoding, the method of ranking of genetic operators and the controlled selection of population members.

Keywords: fuzzy control, multistage optimal fuzzy control, adaptive evolutionary algorithm.

1. Introduction

Fuzzy logic is a generalization of classical, binary logic to rules and statements, which cannot be valued as just true or false. Similarly, it is used to describe situations and effects where exact values are not necessary, but qualitative evaluations with a very small set of possible values (i.e. small, medium, big) are sufficient. Nowadays it is one of the most developed approaches, which enables

reasoning based on imprecise or incomplete information. Fuzzy logic makes it possible and easy to decide when facts and rules are uncertain. This uncertainty may not be caused by inaccuracy of measurements or too small level of knowledge about the problem but it can be its immanent property. This powerful apparatus was invented by L.A. Zadeh (Zadeh, 1965), who introduced basic ideas and theoretic foundations of fuzzy sets and fuzzy logic. Since then the domains of fuzzy sets and fuzzy logic made a great development and became a leading part of artificial intelligence. Some more interesting information about fuzzy logic and its applications can be found in Czogała, Pedrycz (1985), Kacprzyk (1997), Piegat (1999).

The most important practical applications of fuzzy logic are in automatic control, where it is widely used, from control of trains and ships to control of photo cameras, TV-sets and other "intelligent" electronic equipment.

The multistage optimal fuzzy control is quite a different approach to the problem of fuzzy control than that presented in Zadeh (1965) and for the first time it was described by Bellman and Zadeh (1970). The principles of both approaches are presented below.

The widely used fuzzy controllers contain control rules, which are fuzzy and imprecise, but the controlled system requirements and limitations are rather well defined. The fuzziness of these rules is caused by the fact that accurate rules or formulae describing the object are unknown, complicated or too difficult to derive and thus it is difficult to find a traditional, model based method to control such object. Fuzzy rules are good, simple and effective estimations of the ideal (or optimal) ones. Fuzzy rules are often based on the empirical knowledge of the problem and can be acquired from experts. Generally, fuzzy controllers behave more like human operators - they do not know the dynamics of the controlled object, but know what to do in order to obtain expected results.

The multistage optimal fuzzy control deals with rather known models and more or less precise control algorithms but, contrary to the approach outlined before, considers fuzzy constraints, requirements, goal functions and often probabilistic or fuzzy models of controlled objects. This approach allows for finding more precise or even optimal controls, but in the sense of fuzzy optimization (with fuzzy constraints, goal functions or models).

The domain of practical applications of this method is quite different (so far) from that of traditional fuzzy control and covers planning, decision making and scheduling, for instance flood prevention in Esogbue, Theologiu and Guo (1992) and Kacprzyk (1997), planning of social and economical regional development in Kacprzyk (1984) or a schedule of power unit switching in Su and Hsu (1991).

The domain of multistage fuzzy control is often divided into several classes depending on the type of object under control: deterministic, stochastic and fuzzy or depending on the control termination time: fixed (specified in advance), specified as a set of termination states of the controlled object, fuzzy, and infinite. Thus, several possibilities of the controlled object type and the control termination time may be derived.

This paper deals with the problem of multistage fuzzy control and finding the optimal control sequence with fixed termination time for the simple deterministic, stochastic and fuzzy objects with finite sets of states, possible controls and known, stationary transition functions. There are also fuzzy constraints imposed on controls and states of the object at every stage. The goal of control is to maximize the quality criterion (the resultant membership function - μ_D) during the time of the particular system's operation.

Conventional methods of solving the problem of multistage optimal fuzzy control cover dynamic programming, branch and bound, interpolative reasoning and also neural nets and genetic algorithms as presented in Kacprzyk (1997). The methods mentioned above are mainly open-loop control methods, but the first steps in applying closed-loop (or feedback) methods in that domain have also been made in Sousa and Kaymak (2001).

Because of several limitations of conventional methods (especially for large dimensions of solved problems), usually used to obtain the optimal set of controls, a specialized evolutionary algorithm was proposed to solve this problem (within the traditional open-loop approach). The problem, the evolutionary algorithm and the obtained results are in details discussed in this paper.

2. Formulation of the multistage fuzzy control problem

2.1. The deterministic system

To solve the problem it is necessary to find a sequence of N (N - control horizon) controls $u(t) \in U$, where U is a finite set of possible discrete controls. The dynamics of the deterministic object under control is characterized by a transition function:

$$x(t+1) = f(x(t), u(t)) \tag{1}$$

given as a matrix of transitions between states $x(t) \in X$ (X is a finite set of possible states of the object). The initial state $x(0)$ is known and fixed. There are fuzzy constraints imposed on controls at each control stage t and $\mu_{C(t)}$ is a membership function of the fuzzy set of admissible controls $C(t)$. Also, fuzzy goals for the consecutive states of the system are defined by membership functions $\mu_{G(t)}$ of a set of desired states $G(t)$. Therefore, the solution of the problem can be found by maximizing the expression (Kacprzyk, 1997):

$$\begin{aligned}\mu_D(\hat{u}(0), \ldots, \hat{u}(N-1)|x(0)) = \max_{u(0),\ldots,u(N-1)} [&\mu_{C(0)}(u(0)) \wedge \mu_{G(1)}(x(1)) \wedge \ldots \\ &\wedge \mu_{C(N-1)}(u(N-1) \wedge \mu_{G(N)}(x(N))]\end{aligned} \tag{2}$$

where:
$\hat{u}(0), \ldots, \hat{u}(N-1)$ - the sequence of optimal controls;
$u(0), \ldots, u(N-1)$ - controls in the consecutive time stages;

$x(0), \ldots, x(N)$ - states of the system in the consecutive time stages;
$\mu_D(..)$ - an overall goal function for the problem;
N - the termination time (control horizon);
$\mu_{G(t)}(..)$ - the membership function of fuzzy set of states ($G(t)$);
$\mu_{C(t)}(..)$ - the membership function of fuzzy set of desired controls ($C(t)$);
$\wedge$ - a t-norm[1] symbol (in the considered case - a minimum function).

The problem described above can be solved using many techniques: dynamic programming - Baldwin and Pilsworth (1982), branch and bound - Kacprzyk (1978), interpolative reasoning - Kacprzyk (1993), neural nets - Francelin and Gomide (1993) and also genetic algorithms[2] in Kacprzyk (1997), Kacprzyk (1998), Kacprzyk, Romero and Gomide (1999), but with the increasing number of control steps, their applicability decreases considerably. The number of possible solutions to the problem, assuming fixed initial state, is equal:

$$n = |U|^N \tag{3}$$

where:

$|U|$ - number of possible controls;
N - termination time;
n - number of possible solutions.

Conventional methods, like dynamic programming and branch and bound, suffer from the effects of growth of the solved problem's dimensionality. They are very good for small values of N and are able then to find optimal solutions. For larger N, only heuristic, random and artificial intelligence based methods can deal with the problem effectively. The most promising ones are neural networks - Francelin and Gomide (1993), genetic and evolutionary algorithms - Kacprzyk (1996), Stańczak (2001) and Stańczak (2003). Unfortunately, genetic and evolutionary algorithms do not ensure finding of the optimal solution in finite time, but they find very good, sub-optimal solutions quite fast. In practical cases it is better to obtain sub-optimal solution quickly than to wait for the optimal one very long time. Thus it seems useful to apply the evolutionary algorithm, designed specially to solve that kind of problem. The approach described brings together some advantages of the evolutionary algorithm with some heuristic and problem-specific methods used as "intelligent" genetic operators.

[1] t-norm and its conorm - s-norm are a generalization of binary logic operators: $\wedge$ - *and* and $\vee$ - or. Fuzzy logic defines its own operators and there are many variations of them with different advantages and disadvantages, Piegat (1999). This paper deals with the simplest and the most widely used case, where $\wedge$ means minimum (...), and $\vee$ maximum (...).

[2] The difference between the notions of "genetic algorithm" and "evolutionary algorithm" is assumed in this paper as follows. Genetic algorithm is a simple method with binary encoding of solutions, traditional mutation, crossover and roulette selection, while evolutionary algorithm is a more general notion, covering a wide range of methods with improved or specialized encoding, operators and selection, self-adaptation of parameters and similar extensions.

2.2. The stochastic system

In the case of the stochastic system the transition function relies on conditional probabilities of accessing some state, depending on a previous state and control signal and is a matrix of probabilities (exactly a set of $|U|$ matrices, depending on used controls). The state of the system is described by a column vector of probabilities - $P(\bar{x}(t))$ (not an actual value of the state, which is unknown and may be expressed only as a probability vector) which contains probabilities of attaining elements (states) of the set X in the moment t

$$P(\bar{x}(t+1)) = P(\bar{x}(t)T(\bar{x}(t+1)|\bar{x}(t)) \tag{4}$$

where:
$P(\bar{x}(t))$, $P(\bar{x}(t+1))$ - probability distribution on the vector of the system state in following iterations;
$T(\bar{x}(t+1)|\bar{x}(t))$ - a matrix of conditional probabilities (transition matrix);
$u(t)$ - deterministic control signal in moment t;
$\bar{x}(t)$, $\bar{x}(t+1)$ - states of the system in the consecutive moments.

The described system is an example of a Markov chain.

Starting from the initial state $\bar{x}(0)$ the system goes towards the terminating state $\bar{x}(N)$, according to (4) with fuzzy constraints imposed on states and controls in each iteration. The choice of proper control signals $u(t)$ enables to maximize the quality criterion (Kacprzyk, 1997):

$$\mu_D(\hat{u}(0),\ldots,\hat{u}(N-1)|x(0)) = \max_{u(0),\ldots,u(N-1)}[\mu_{C(0)}(u(0)) \wedge E\mu_{G(1)}(\bar{x}(1)) \wedge \ldots \wedge \mu_{C(N-1)}(u(N-1)) \wedge E\mu_{G(N)}(\bar{x}(N))] \tag{5}$$

where: $E\mu_{(G(t)}(\ldots)$ - expected value of the membership function of set $G(t)$ - a fuzzy constraint (or fuzzy goal) imposed on state of the system, defined as:

$$E\mu_{G(t)}(\bar{x}(t)) = \sum_{i=1}^{|X|} p(x_i(t))\mu_{G(t)}(x_i(t)) \tag{6}$$

where:
$p(x_i(t))$ - probability that the system is in state $x_i \in X$ or the i-th coordinate of vector $P(\bar{x}(t))$;
$\mu_{G(t)}(x_i(t))$ - a value of the membership function $G(t)$ for the state $x_i \in X$;
all other symbols like in (2) and (4).

Methods usually used to solve the problem are similar to those described in Section 2.1. It means that also methods based on dynamic programming, interpolative reasoning and neural nets were proposed (Kacprzyk, 1997), but they are properly adjusted to deal with the problem of stochastic systems. The size of the space of solutions can be estimated from formula (3). As it dramatically grows with the time horizon of control, it is natural to propose a specialized evolutionary algorithm to solve it, especially for larger values of N.

2.3. The fuzzy system

The fuzzy system is described with fuzzy state equation:

$$X(t+1) = f(X(t), U(t)) \tag{7}$$

where:

$X(t)$, $X(t+1)$ - fuzzy states of the system in moments t and $t+1$ belonging to the finite set of allowable fuzzy states $\mathcal{X}$;

$U(t)$ - control signal (fuzzy or deterministic) in the moment t, from the finite set of possible controls $\mathcal{U}$.

This relationship can be more precisely rewritten using membership functions of sets of states and controls (when necessary). The formula (8) shows the situation where deterministic controls are applied, the formula (9) describes the fuzzy system with fuzzy control (Kacprzyk, 1997):

$$\mu_{X(t+1)}(x_i(t+1)) = \max_{j \in 1..|\mathcal{X}|} (\mu_{X(t)}(x_j(t)) \wedge \mu_{X(t+1)}(x_i(t+1)|x_j(t), u(t))) \tag{8}$$

$$\mu_{X(t+1)}(x_i(t+1) = \max_{j \in 1..|\mathcal{X}|} [\max_{k \in 1..|\mathcal{U}|} (\mu_{X(t)}(x_j(t)) \wedge \mu_{X(t+1)}(x_i(t+1)|x_j(t), u_k(t) \wedge \mu_{U(t)}(u_k(t)))] \tag{9}$$

where:

$\mu_{X(t)}(x_i(t+1))$ - membership function value of coordinate x_i of the controlled system state X in the moment $t+1$;

$\mu_{X(t+1)}(x_i(t+1)|x_j(t), u(t))$ - conditional membership function, which describe transitions between states $X(t)$ and $X(t+1)$ under control $u(t)$, in the considered case a matrix of transitions between states, containing values of membership functions;

$u_k(t)$ - represents the k-th coordinate of fuzzy control $U(t)$;

all other symbols like in the formula (7).

Similarly as in the case of deterministic and stochastic systems, there are fuzzy constraints imposed on states ($\mu_{G(t)}$) and controls ($\mu_{C(t)}$) of the system in every iteration. The initial state of the system is known and fixed ($X(0)$).

To solve the problem a tool for comparing fuzzy sets is needed. The method is described in Kacprzyk (1997) and Kacprzyk (1996) and is called similarity function:

$$\mu_{\bar{C}(t)}(U(t)) = 1 - d(U(t), C(t)) \tag{10}$$

$$\mu_{\bar{G}(t)}(X(t)) = 1 - d(X(t), G(t)) \tag{11}$$

where:

$d(..)$ - distance (Hamming[3], Euclidean or different);

[3]In the considered case Hamming distance was applied in computer simulations.

$C(t)$ - set of constrains imposed on controls in iteration t;
$G(t)$ - set of constrains imposed on state in iteration t;
$\mu_{\bar{C}(t)}(U(t))$ and $\mu_{\bar{G}(t)}(X(t))$ - measure of similarity of membership functions.

A quality criterion for the system with fuzzy control is shown by the formula (12), with deterministic control by the formula (13) (Kacprzyk, 1997):

$$\mu_D(\hat{U}(0),\ldots,\hat{U}(N-1)|X(0)) = \max_{U(0),\ldots U(N-1)}[\mu_{\bar{C}(0)}(U(0)) \wedge \mu_{\bar{G}(1)}(X(1)) \wedge \ldots \wedge \mu_{\bar{C}(N-1)}(U(N-1)) \wedge \mu_{\bar{G}(N)}(X(N))] \quad (12)$$

$$\mu_D(\hat{u}(0),\ldots,\hat{u}(N-1)|X(0)) = \max_{u(0),\ldots,u(N-1)}[\mu_{\bar{C}(0)}(u(0)) \wedge \mu_{\bar{G}(1)}(X(1)) \wedge \ldots \wedge \mu_{\bar{C}(N-1)}(u(N-1)) \wedge \mu_{\bar{G}(N)}(X(N))] \quad (13)$$

where:
$\hat{U}(0),\ldots,\hat{U}(N-1)$ - the sequence of optimal fuzzy controls;
$U(0),\ldots,U(N-1)$ - fuzzy controls in subsequent moments;
$\hat{u}(0),\ldots,\hat{u}(N-1)$ - the sequence of optimal deterministic controls;
$u(0),\ldots,u(N-1)$ - deterministic controls in subsequent time stages;
$X(0),\ldots,X(N)$ - states of the system in subsequent iterations;
$\mu_D(..)$ - an overall goal function for the problem;
N - termination time;
$\wedge$ - t-norm sign;
all other symbols like in the formulae (10) and (11).

Similarly to previously described stochastic and deterministic systems, the methods of solving the described problem cover dynamic programming, branch and bound, interpolative reasoning, neural nets and also genetic algorithms. In the case of the deterministic controls, the solution space can be described by equation (3), for fuzzy controls the solution space is continuous and all possible solutions are contained in a hypercube $[0,1]^{N|U|}$. A high number of dimensions of the problem makes it difficult to solve and it will be shown that the evolutionary algorithm is a good choice to solve that problem.

3. The evolutionary algorithm in the problem of optimal fuzzy control

The evolutionary algorithm can be described as follows:

1. Random initialization of the population.
2. Reproduction and modification of solutions, using genetic operators.
3. Evaluation of the obtained solutions.
4. Member selection for the next generation.
5. Unless the stop criterion is satisfied, go to the point 2.

In spite of their universality, the evolutionary algorithms should be properly adjusted to solve a raised problem. Unchanged remains only the core idea

of its functioning: owing to small random changes in genotypes of population members (mutation), the recombination of genes and the selection of the best individuals, the population develops towards better values of the problem's goal function. The adjustment of the genetic algorithm to the solved problem requires a proper encoding of solutions and an invention of proper genetic operators for that problem. Genetic evaluations stop after a fixed number of generations, when satisfying results are obtained or there are no changes of the fitness function during some number of generations.

3.1. The individual encoding for EA

3.1.1. The case of deterministic controls

Deterministic controls can be used in all kinds of previously described systems. In the case of optimal control problem, the whole information about the current solution is stored in a sequence of controls (Kacprzyk, 1997) (assuming a fixed initial state). Thus the genotype of every member of the population is a sequence of indexes (integer values) of control signals from the set U for the subsequent steps of the controlled object operation. This method of solution encoding is very effective and gives the advantage of elimination of faulty solutions. A real member of the population, used for evolutionary computations, contains an integer vector of N values of controls. Some data required by the evolutionary algorithm are also added: a number of a genetic operator chosen to modify the solution in the current iteration and a vector of qualities of genetic operators, which is a base to choose one in every iteration. Fig. 1 presents a scheme of an individual of the population with deterministic controls.

3.1.2. The case of fuzzy controls

To the fuzzy system, described by equations (7), (8) and (12), fuzzy controls are applied. Fuzzy controls are vectors of real values. These real numbers are the values of the membership function of the set of allowable controls U. Different values of these real numbers (different membership functions) constitute different control signals.

This implies that a member of the population contains N (number of control steps) vectors of membership function values, thus it is a matrix $Nx|\mathcal{U}|$ of real values from the interval $[0, 1]$. This method of encoding also ensures that only solutions satisfying requirements of the problem may appear. Similarly as in the case of deterministic controls a member of the solution population contains also the number of the chosen genetic operator and the vector of qualities of operators. The method of operator selection is precisely described in Section 3.4. Fig. 2 presents the scheme of an individual from the population with fuzzy controls.

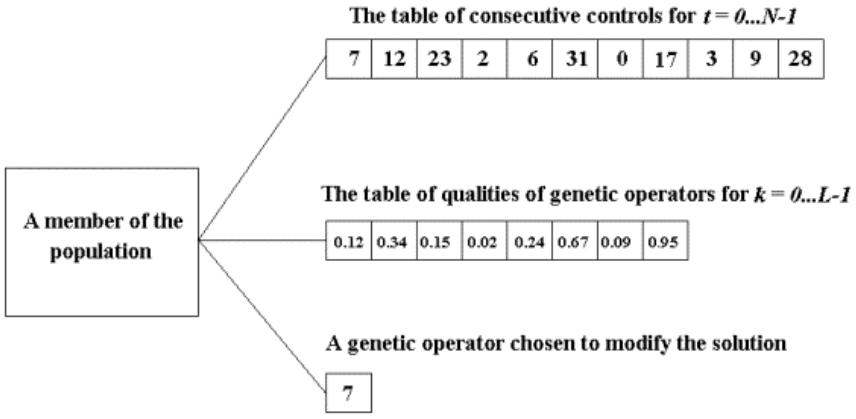


Figure 1. A member of the population of solutions for the case of deterministic controls

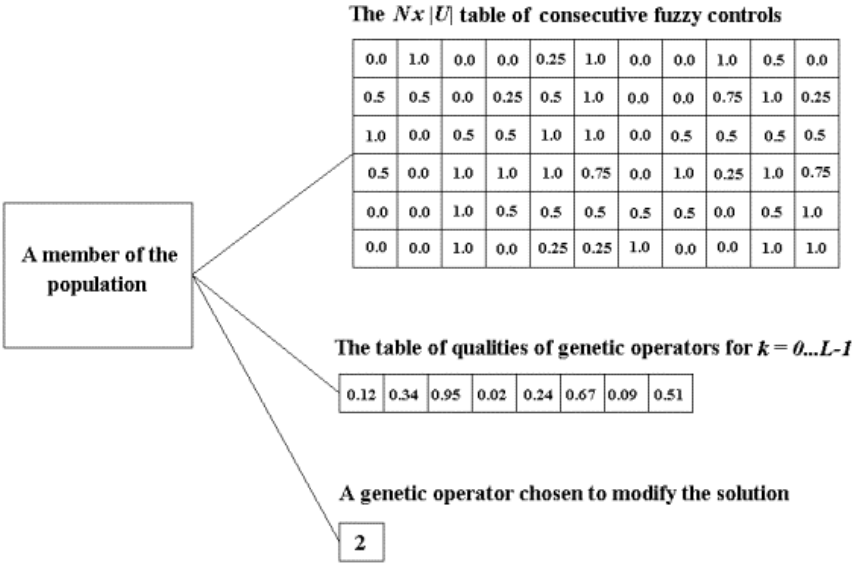


Figure 2. A member of the population of solutions for the case of fuzzy controls

3.2. The fitness function

In all the cases considered, the quality functions of the corresponding problems (2, 5, 12 or 13) have appropriate properties for being directly treated as fitness functions for an evolutionary algorithm:

- non-negative values of membership functions are from the interval $[0, 1]$;
- the better the quality of the individual, the higher the value of the problem's quality function (maximization);
- fuzzy constraints imposed on states and controls are a part of the quality function, so it is easy to consider them and no additional elements are necessary.

Thus, problem's quality functions are used as fitness functions for EA in the here considered problem of optimal fuzzy control.

3.3. Genetic operators

The described data structures require specialized genetic operators, which modify the population of solutions. Simple random operators are easy to devise for both methods of encoding (with integer and real numbers). They are similar to the widely used ones: mutation, inversion and crossover. Also multiple versions of described operators can be here applied. Simulation of the evolutionary algorithm with such "blind" operators proved that the more sophisticated ones, enriched with some heuristics, should be used. Random operators have difficulties with finding even very poor but non-zero solutions (Stańczak, 2001), especially for problems with deterministic control or $N > 10$.

It is obvious that heuristic operators give only a higher possibility of finding better solutions than random ones, but not a 100% certainty. Several versions of heuristic operators were used together with random operators during computer simulations. Application of rather big number of genetic operators (about 12) caused some problems with the choice of appropriate probabilities of operators' appearance. The method used for this purpose is presented in Section 3.4.

3.3.1. Genetic operators for deterministic control

Three standard, random operators were applied:

- crossover - an exchange of randomly found last parts of encoding strings;
- mutation - a random change of one randomly selected control to another from the set U;
- inversion - an inversion of a randomly chosen string subset of the solution.

Besides, the same operators appeared in a multiple version - 3 repetitions of the operator. These operators belong to the class of "blind", random operators. They are specialized to use accepted solution encoding, but do not know anything about the solved problem. As it was previously mentioned, experiments

with such a set of operators gave rather poor results (Stańczak, 2001), especially for $N > 10$.

Three heuristic operators for deterministic control have been devised:

- "intelligent" mutation;
- "forward" genetic operator;
- "backward" genetic operator.

The first of the heuristic operators resembles mutation but with some heuristics added, so as to be called "intelligent" mutation. Traditional mutations works like the operator not on randomly chosen bits of the solution. In this case mutation is more complicated. It works as follows: in a randomly chosen place of the encoding string, it changes one control to another one, randomly chosen from the set U. "Intelligent" mutation acts in almost the same way, but the new control is randomly chosen from the set of admissible controls (i.e. with $\mu_{C(t)}(t) > 0$) for the current step t. Sets of admissible controls for every step of the system are created during the initialization of the EA, and are also used in other heuristic operators. This way of generating new solutions is potentially more effective than the simple mutation and gives a higher probability of positive change of modified solution.

The second of them is called "forward", because it follows control signals from $t = 0$ to $t = N - 1$. The algorithm of the operator is shown in Algorithm 1. All symbols in Algorithms 1 and 2 have the same meaning as in Section 2.1. The presented notation of the Algorithm 1 concerns the case of deterministic system under control, but it can easily be transformed to all cases of system with deterministic control.

For $i := 1$ to N do
Begin
  Evaluation of $\mu(i)$ (a local membership function value):
    $\mu(i) := \mu_{C(i-1)}(u(i-1)) \wedge \mu_{G(i)}(x(i))$;
    where: $x(t) = f(x(t-1), u(t-1))$
  If $\mu(i) \leq \mu_D(u(0), , u(N-1)|x(0))$
    a) creation of the list of admissible pairs
      $(u_l(i-1), x_l(i))$, $l \in 1, , |U|$, where:
      $x_l(t) = f(x(t-1), u_l(t-1))$
      $\mu_l(i) := \mu_{C(i-1)}(u_l(i-1)) \wedge \mu_{G(i)}(x_l(i))$
      and
      $\mu_l(i)\mu_D(u(0), , u(N-1)|x(0))$;
      $x(t-1)$ - the previous state of the system;
    b) selection of one pair from the list (several choice methods are possible);
    c) modification of the solution on the position i (if the list of possible pairs is empty, the position i is left unchanged and the operator stops);
End;

Algorithm 1. The algorithm of the "forward" genetic operator

The "forward" operator tries to correct the places, where the local membership function in the pre-defined trajectory drops to 0 or to the lowest value and consequently the overall quality function (which is a minimal value of local membership functions values) is equal 0 or this lowest value. It is done by checking all other possible controls instead of the previously used in that iteration in order to find the better ones. If this succeeds, a list of possible better solutions is made. Then, one of the controls from the list is chosen[4] and replaces the formerly used one. If the list for some value of t is empty (it may happen in some circumstances) no operation is performed and the operator ends its work.

The last heuristic operator is called "backward", because it works similarly to the "forward" one but in the reverse order (from $t = N - 1$ to $t = 0$). In the case of control (or the position in the solution) when a too low value of the membership function is found, an appropriate control is chosen (like in the case of "forward" operator) from the list of possible controls and state. This chosen control replaces the previous one. The operator produces the list of controls for potential replacement. The control is selected from the list so as to preserve the sequence of the earlier established states of the controlled object. Similarly like in the "forward" operator, it may happen that no control can be selected to satisfy constraints. In that situation the operator stops for $t > 0$, but changes performed before that point are valid. It is often difficult to hit the fixed starting state, because for chosen $x(1)$, $u(0)$ and fixed $x(0)$ there is only one possibility for $x(1)$ to be equal $f(x(0), u(0))$ or not and no other moves are allowed. In such a case the operator does not rather improve the solution but works like some kind of random genetic operator. The detailed procedure of the method is presented in Algorithm 2.

For $i := N$ downto 1 do
Begin
  Evaluation of $\mu(i)$ (a local membership function value)
    $\mu(i) := \mu_{C(i-1)}(u(i-1)) \wedge \mu_{G(i)}(x(i))$
  for all possible pairs $(u_l(i-1), x_k(t))$:
    if $i = N$: $l \in 1,,|U|$ and $k \in 1,,|X|$
    if $i \neq N$: $l \in (1,,|U|$ and k is fixed - selected in the previous step;
  Selection of one pair such that:
    $\mu(i) > \mu_D(u(0),,u(N-1)|x(0))$
  Selection from the transition matrix of a state $x(i-1)$, for which:
    $x(i) := f(x(i-1), u(i-1))$;
  Modification of the i-th position of the solution with chosen control

[4]Several methods of selection of controls from the list were tested and applied: random, best one and scaled random (roulette). In computations, all three methods are used randomly to increase the possibility of finding better sequences of controls.

(if no control is selected or state $x(i-1)$ cannot be attained, solution on position i remains unchanged and operator stops);
End;

Algorithm 2. The algorithm of the "backward" genetic operator

The "backward" genetic operator can be successfully applied only in the case of the deterministic system, because it is then easy to find an inverse transition function. In the case of a stochastic system an inverse matrix of probabilities is required to implement the operator, but an inverse simulation of the system operation may give states with negative or greater than 1 "probabilities" in the state vector. This is caused by numerical errors and the fact that inverse matrices contain also values negative or greater than 1. So, in the case of a stochastic system its usefulness is rather problematic and it has not been tried out in simulations. For the fuzzy system it is impossible to perform the inverse simulation of the system operation and the "backward" operator cannot work.

3.3.2. Genetic operators for fuzzy control

A real value matrix encoding requires the use of a different set of genetic operators than in the case of integer vectors. They are similar to methods used in evolutionary strategies but adjusted to matrix representation used in the solved problem:

- mutation - a small random change of the randomly chosen element of the solution, which precisely means a small modification of one value of the membership function, not exceeding the range $[0, 1]$;
- transposition - an exchange of randomly chosen vectors of control membership functions in the same individual;
- crossover - an exchange of control vectors between two individuals, where the point of crossing is obtained randomly;
- inversion - an inversion of a randomly chosen subset of vectors of the solution.

Also a heuristic operator was added. Its operation resembles a little the manner in which the "forward" heuristic operator works.

For $i := 1$ to N do
Begin
Evaluation of $\mu(i)$ (a local membership function value):
$\mu(i) = \mu_{\bar{C}(i-1)}(U(i-1)) \wedge \mu_{\bar{G}(i)}(X(i))$
If $\mu(i) \leq \mu_D(U(0), ..., U(N-1)|X(0))$ then:
a) if $\mu_{\bar{C}(i-1)}(U(i-1)) \leq \mu_D(U(0), ..., U(N-1)|X(0))$ then:
- $U_p(i-1) = C(i-1)$;
- $\mu_p(i) = \mu_{\bar{C}(i-1)}(U_p(i-1)) \wedge \mu_{\bar{G}(i)}(X_p(i))$;
- if $\mu_p(i) > \mu(i)$ then $U(i-1) = U_p(i-1)$, $X(i) = X_p(i)$
else Exit;

b) if $\mu_{\bar{G}(i)}(X(i)) \leq \mu_D(U(0), ..., U(N-1)|X(0))$ then:
- $U_p(i-1) = (|\mathcal{X}| * C(i-1) + |\mathcal{U}| * G(i))/(|\mathcal{X}| + |\mathcal{U}|)$;
- $\mu_p(i) = \mu_{\bar{C}(i-1)}(U_p(i-1)) \wedge \mu_{\bar{G}(i)}(X_p(i))$;
- if $\mu_p(i) > \mu(i)$ then $U(i-1) = U_p(i-1)$, $X(i) = X_p(i)$ else Exit;

End;

Algorithm 3. The algorithm of the heuristic operator for problem with fuzzy control

The operator follows the sequence of control vectors from $t = 0$ to $t = N-1$. During that time it finds places where the local membership function ($\mu(i)$) drops to the lowest values and tries to increase them by changing the values of the membership functions in the control vector. The values are changed to make the vectors of fuzzy states and controls closer to fuzzy constraints imposed on the system in the current time step.

The algorithm of the heuristic operator for fuzzy control is shown in Algorithm 3 (all symbols like in formulae 7, 8 and 12).

3.4. Evolutionary algorithm with ranking of genetic operators

Using rather a big number of genetic operators requires applying some method of sampling them in all iterations of the algorithm. In the used approach, based on Mulawka and Stańczak (1999), Stańczak (1999, 2000, 2003), it is assumed that an operator that generates good results should have bigger probability of application and should more frequently affect the population. But it is very likely that an operator, which is good for one individual, gives worse effects for another one, for instance because of its location in the domain of possible solutions. Thus, every individual may have its own preferences. Every individual has a vector of floating point numbers - q (besides the encoded solution). Each position in the vector q corresponds to one genetic operation. It is a measure of quality of the genetic operator. The probability of execution of the operator is proportional to the value of its quality. This relationship may be written as follows:

$$p_{ij}(t) = \frac{q_{ij}(t)}{\sum_{i=1}^{L(t)} q_{ij}(t)} \tag{14}$$

where:
$q_{ij}(t)$ - quality coefficient of the operation i at the moment t for the member j;
$p_{ij}(t)$ - probability of appearance of the operation i at the moment t for the member j;
$L(t)$ - number of genetic operators (may vary during genetic computations).

This ranking becomes the basis for computing the probabilities of appearance and execution of genetic operators. This set of probabilities is also a base

of experience of every individual and according to it an operator is chosen in each epoch of the algorithm. Due to the accumulated experience one can maximize the chances of its offspring to survive. The quality factors are computed according to the formula (15):

$$q_{ij}(t+1) = \begin{cases} q_{lj}^0(t) + x_{lj}(t) + \alpha_{lj}(t) * q_{lj}(t) & \text{for i=l} \\ q_{ij}(t) & \text{otherwise} \end{cases} \tag{15}$$

where:
$q_{ij}(t+1)$, $q_{ij}(t)$ - quality of the operation i for the individual j in consecutive generations;
$q_{ij}^0(t)$ - a small credit value;
$x_{lj}(t)$ - an improvement of the problem's quality function, obtained by the l-th operator for the j-th member of population, defined as follows:

$$x_{ij}(t) = \begin{cases} Q(t) - Q_{ij}(t) & \text{minimization} \\ Q_{ij}(t) - Q(t) & \text{maximization} \\ 0 & \text{no improvement} \end{cases} \tag{16}$$

where:
$Q(t)$, $Q_{lj}(t)$ - the best found solution value and the value of solution for member j;
$\alpha_{lj}(t)$ - a coefficient of forgetting, $\alpha \in [0, 1]$;
l - an index of the operator chosen for modification of the particular solution.

The first element of the formula (15) - $q_{ij}^0(t)$ - plays the role of a credit, a small value, which supports a small level of $q_{ij}(t)$ even if the operator does not give any advantages for a long term. Dropping this value to zero would eliminate the operation corresponding to it for current individual and for its possible offspring. This fact is not advantageous, because it is possible that the excluded operator will work better on other stages of the evolution process. For exploring operators, like mutation, it is often necessary to let them work even without any visible improvements of the fitness function.

The second addend is an improvement of the problem's quality function in the current generation or zero when no improvement is achieved.

The third part of the formula (15) is responsible for storing previous achievements of an operator multiplied by the forgetting factor $\alpha_{lj}(t)$. It is responsible for balancing the influence on the quality factor of an operator originating from old and new improvements. Decreasing the value achieved by an operator is introduced to make the evolutionary algorithm more flexible. It should be noticed that some genetic operators may achieve good results in some phases of simulation, and then exhaust their capacities, but other ones, probably better in the later phases, would have small probabilities of appearance, so it would take a lot of time to change this situation. The effect of forgetting former achievements can overcome this problem. When operators do not change the global

best solution for some time, the probabilities of operators become almost equal. After every generation only the value $q_{ij}(t+1)$ associated with the chosen operator is updated, the other ones remaining unchanged. Only one operator is executed in one generation for one individual, thus there is no reason to change the coordinates corresponding to the other, not selected operators.

3.5. Controlled selection

The applied selection method consists of two methods with different properties: histogram selection (increases the diversity of the population) and deterministic roulette (strongly promotes best individuals) (Stańczak, 1999), which are selected at random during the execution of the algorithm. The probability of execution of the selection method is obtained from the formula (17).

If individuals in the population are described by a too small standard deviation of the fitness function ($\sigma(F(t))$) with respect to the extent of this function ($\max(F_{\mathrm{av}}(t)-F_{\min}(t), F_{\max}(t)-F_{\mathrm{av}}(t))$ or $R(t)$), then it is desirable to increase the probability of appearance of histogram selection. In the contrary situation the probability of the deterministic roulette selection is increased. If the parameters of the population are located in some range, considered advantageous, we may keep approximately the same probabilities of appearance for both methods of selection. It is important that always $p_{\mathrm{his}}(t)+p_{\mathrm{det}}(t)=1$, which means that some method of selection must be executed.

$$p_{\mathrm{his}}(t+1)=\begin{cases} p_{\mathrm{his}}(t)(1-a) & \text{if} \quad R(t)>3\sigma(F(t)) \\ p_{\mathrm{his}}(t)(1-a)+a & \text{if} \quad R(t)<0.5\sigma(F(t)) \\ p_{\mathrm{his}}(t)(1-a)+0.5a & \text{if} \quad R(t)\geq 0.5\sigma(F(t)) \\ & \quad \wedge R(t)\leq 3\sigma(F(t)) \end{cases} \tag{17}$$

where:

$p_{\mathbf{his}}(t)$ - probability of application of histogram selection ($1-p_{\mathrm{his}}(t)$ is the probability of application of deterministic roulette method $p_{\mathrm{det}}(t)$);

$R(t)=\max(F_{\mathbf{av}}(t)-F_{\min}(t), F_{\max}(t)-F_{\mathbf{av}}(t))$;

$F_{\mathbf{av}}(t)$, $F_{\min}(t)$, $F_{\max}(t)$ - average, minimal and maximal values of fitness function in the population;

$\sigma(F(t))$ - standard deviation of fitness function value in the population;

a - a small value to change probability $p_{\mathrm{his}}(t)$, in simulations $a=0.05$.

4. Simulation results

Results, obtained using the simple genetic algorithm (Kacprzyk, 1996, 1997; Kacprzyk, Romero and Gomide, 1999), were promising, but the evolutionary methods were expected to behave better, especially for a bigger number of control steps. To overcome this problem it is necessary to use the specialized evolutionary algorithm, which was described in the previous section. Results of

evolutionary evaluations are presented in this section. Common parameters of computer simulations conducted are listed below:

- the results presented are minimal/average/maximal values obtained from 100 simulations for each problem, in cases where a smaller number of simulations were conducted (due to a long time of simulation) this number is shown;
- the presented values are computed using best solutions obtained during computations;
- starting populations are generated randomly for the not specialized EA or in the following manner for the specialized EA: half of the population is obtained randomly from the set of all possible controls. The second part is generated randomly but only from the set of controls with the values of membership functions for the current step greater than 0 (which gives better starting points than the simple random method);
- the same problem was solved with and without heuristic operators;
- all solved problems were generated randomly;
- the strategy $(\mu + \lambda)$[5] was applied, where $\mu = 60$, $\lambda = 420$[6];
- for each simulation the same methods of individual selection and ranking of genetic operators were used.

4.1. The deterministic system

In the conducted simulations three sizes of problems were considered, with $N = 10$, 100 and 1000. For all cases $|U| = 32$, $|X| = 20$, with appropriate number of fuzzy constrains for controls and stages for all sizes of the problem. It would be difficult to present all the parameters of the solved problems in this paper, so only a short description of them is presented. The transition function is a $20x32$ matrix that describes which state (one of 20) is obtained from the previous one using one of 32 possible controls. Controls and states are numbers from $[0, 1]$, which is chosen for simplicity and every problem can be scaled to that interval. The problem's goal function (2) is also a fitness function of the evolutionary algorithm. Membership functions for imposed constraints are trapezoidal functions on X and U. For every element of X and U a membership function value is given. These values, ordered from the smallest to the highest value of the support, form a vector, which is a fuzzy constraint $G(t)$ or $C(t)$.

In Figs. 3 and 4 graphical comparisons of obtained results are shown (mean values of best results from 100 simulations). Table 1 contains numerical values of the results presented in Figs. 3 and 4 and additionally the values of the worst and the best solutions obtained at definite time moments during 100 simulations. The comparison of the presented minimal, average and maximal values

[5]The symbol $(\mu + \lambda)$ means that the descendant population is chosen from parents and offspring.

[6]Some authors (Arabas, 2001) claim that there is an optimal proportion between λ and μ: $\lambda/\mu = 7$, which gives the best results and shortest computations.

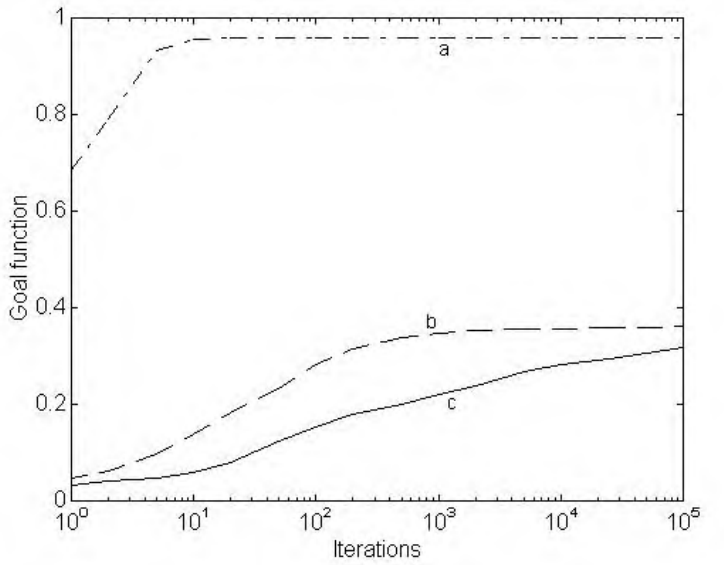


Figure 3. EA with both types of operators: a - N=10, b - N=100, c - N=1000

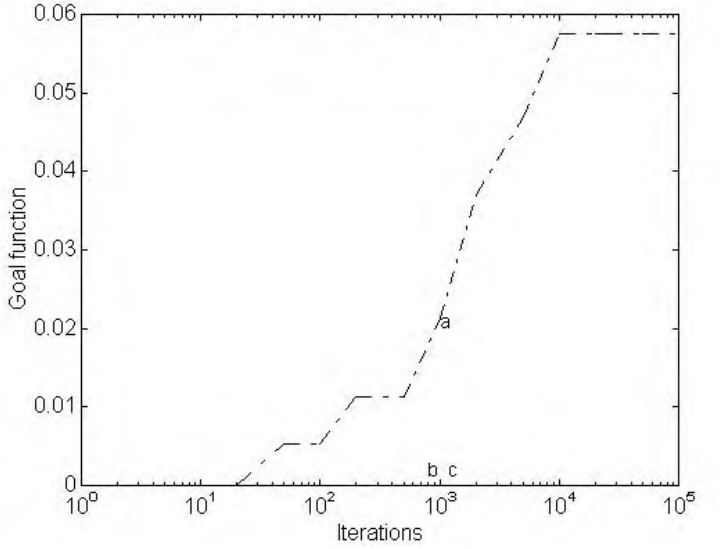


Figure 4. EA with random operators only: a - N=10, b - N=100, c - N=1000

Table 1. Numerical results of simulations for the deterministic system

1. EA with both types of operators											
Iterations	0	1	5	10	50	100	500	1000	5000	10000	50000	100000
(min)	0.000	0.165	0.667	0.891	0.959	0.959	0.959	0.959	0.959	0.959	0.959	0.959
N=10 (av)	0.005	0.685	0.931	0.957	0.959	0.959	0.959	0.959	0.959	0.959	0.959	0.959
(max)	0.165	0.959	0.959	0.959	0.959	0.959	0.959	0.959	0.959	0.959	0.959	0.959
(min)	0.000	0.000	0.017	0.046	0.166	0.166	0.302	0.303	0.323	0.337	0.337	0.337
N=100 (av)	0.000	0.046	0.096	0.138	0.234	0.282	0.338	0.347	0.356	0.357	0.359	0.362
(max)	0.000	0.177	0.258	0.303	0.337	0.376	0.376	0.376	0.376	0.376	0.376	0.376
(min)	0.000	0.004	0.020	0.027	0.088	0.120	0.166	0.182	0.227	0.259	0.279	0.282
N=1000 (av)	0.000	0.032	0.048	0.058	0.123	0.153	0.199	0.219	0.269	0.283	0.306	0.319
(max)	0.000	0.342	0.347	0.347	0.347	0.347	0.347	0.347	0.347	0.353	0.388	0.390
2. EA with random operators												
Iterations	0	1	5	10	50	100	500	1000	5000	1000	50000	100000
(min)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N=10 (av)	0.000	0.000	0.000	0.000	0.005	0.005	0.011	0.021	0.047	0.058	0.058	0.058
(max)	0.000	0.000	0.000	0.000	0.134	0.249	0.653	0.865	0.959	0.959	0.959	0.959
(min)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N=100 (av)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
(max)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
(min)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N=1000 (av)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
(max)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

obtained during genetic computations may be helpful to realize that the results can be very different in simulations conducted with the same parameters. To be exact, the parameters of simulations are always different, because the starting population is generated randomly (with or without additional information about the problem) and is differently distributed in the solution space. It is easy to notice that better results are obtained with the method using both types of operators, especially for higher values of N (100, 1000). The method using random operators and randomly generated starting population gives very poor results. Poor results are connected in this case with a very low probability of random generation of any admissible solution and, as it can be seen, it happened several times only for $N = 10$.

4.2. The stochastic system

In this section, results obtained for the stochastic system are discussed. Evolutionary computations were conducted for $N = 10$, $N = 100$ and $N = 1000$ (for $N = 1000$ computations last very long and so only 15 simulations were performed). Fortunately, in practical cases computations for such a long time horizon are very rare, but are interesting for comparing the properties of algorithms. Similarly to the previously described case, the system with 20 states and 32 possible deterministic controls with appropriate number of fuzzy constraints (constructed in the same manner) for them was assumed. The transition matrix for that case contains probabilities of transitions among states, considering possible controls. Precisely, it is a set of 32 matrices 20x20, depending on applied controls. In Figs. 5 and 6 results are shown graphically, whereas Table 2 presents their numerical values.

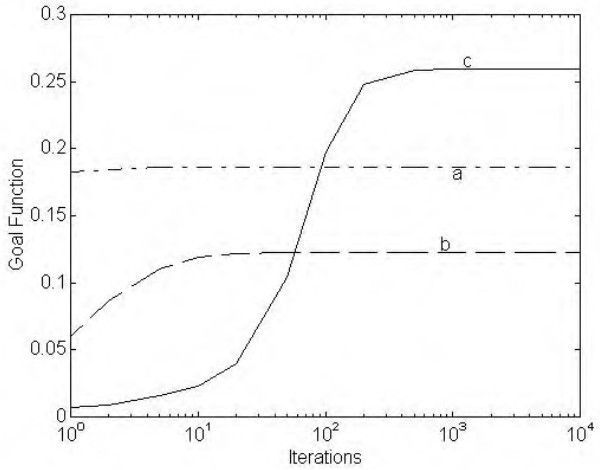


Figure 5. EA with both types of operators: a - N=10, b - N=100, c - N=1000

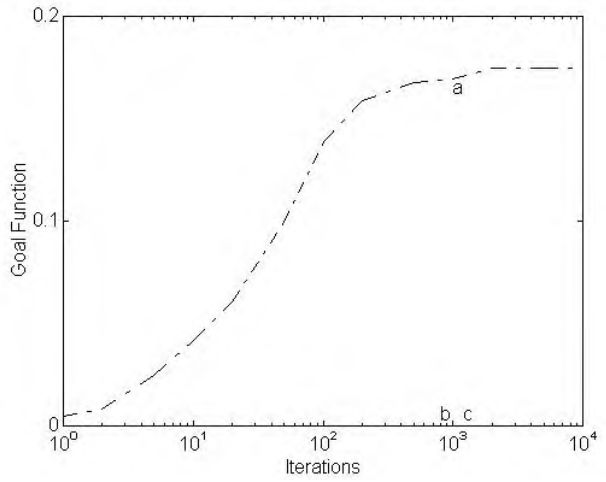


Figure 6. EA with random operators only: a - N=10, b - N=100, c - N=1000

Table 2. Numerical results of simulations for the stochastic system

1. EA with both types of operators										
Iterations	0	1	5	10	50	100	500	1000	5000	10000
(min)	0.148	0.163	0.184	0.186	0.186	0.186	0.186	0.186	0.186	0.186
N=10 (av)	0.173	0.182	0.186	0.186	0.186	0.186	0.186	0.186	0.186	0.186
(max)	0.185	0.186	0.186	0.186	0.186	0.186	0.186	0.186	0.186	0.186
(min)	0.016	0.021	0.081	0.090	0.122	0.122	0.122	0.122	0.122	0.122
N=100 (av)	0.036	0.060	0.110	0.119	0.122	0.122	0.122	0.122	0.122	0.122
(max)	0.088	0.118	0.122	0.122	0.122	0.122	0.122	0.122	0.122	0.122
(min)	0.004	0.005	0.009	0.018	0.069	0.170	0.258	0.259	0.259	0.259
N=1000 (av)	0.006	0.007	0.016	0.023	0.104	0.198	0.259	0.259	0.259	0.259
(max)	0.011	0.011	0.026	0.037	0.130	0.233	0.259	0.259	0.259	0.259
2. EA with random operators										
Iterations	0	1	5	10	50	100	500	1000	5000	10000
(min)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.175	0.175
N=10 (av)	0.000	0.005	0.024	0.041	0.100	0.139	0.168	0.170	0.175	0.175
(max)	0.000	0.173	0.175	0.175	0.175	0.175	0.175	0.175	0.175	0.175
(min)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N=100 (av)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
(max)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
(min)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N=1000 (av)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
(max)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

The results presented in this section also provide the evidence for the proposition that heuristic operators are very powerful. They make it possible to obtain good results even for large dimensions of the solved problem or significantly speed up evolutionary computations for small values of N. The version with random operators only, which resembles a bit a simple genetic algorithm (only simple random operators are used, but other parameters are the same as in the "heuristic" version), works rather poorly.

It should be noticed that the stochastic system needs less iterations than the deterministic one to find good solutions. Unfortunately, though, one iteration lasts much longer and the overall time of evolutionary computations is much longer than in the deterministic case.

4.3. The fuzzy system with deterministic control

The EA for solving the problem of optimal fuzzy control for a fuzzy system with deterministic controls was tested using an object described by a transition function with 20 fuzzy states and 32 deterministic controls. The dynamics of this object is described by 32 matrices 20x20, also fuzzy constraints are imposed on states and controls in a manner similar to the deterministic version. The actual state of the system is described by a vector of membership function values consisting of 20 membership values to elements of X. Tests were performed for $N = 10$, $N = 100$ and $N = 1000$. The results of simulations are presented in Figs. 7 and 8 and Table 3.

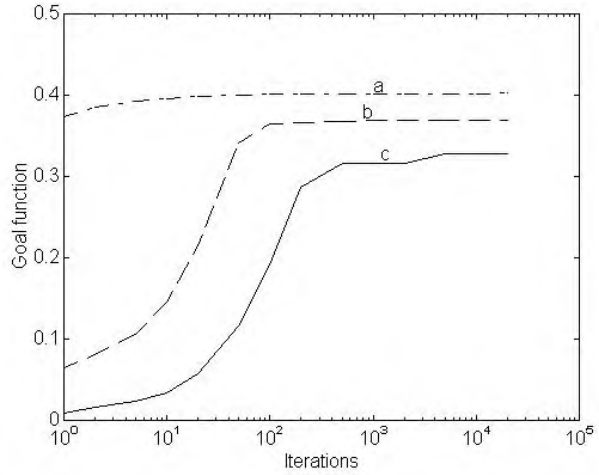


Figure 7. EA with both types of operators: a - N=10, b - N=100, c - N=1000

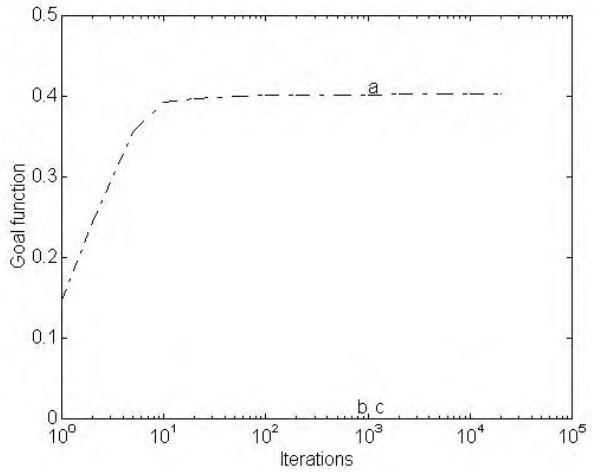


Figure 8. EA with random operators only: a - N=10, b - N=100, c - N=1000

Table 3. Numerical results of simulations for the fuzzy system with deterministic controls

1. EA with both types of operators											
Iterations	0	1	5	10	50	100	500	1000	5000	10000	20000
(min)	0.259	0.361	0.382	0.390	0.393	0.393	0.397	0.398	0.398	0.398	0.398
N=10 (av)	0.358	0.374	0.393	0.396	0.400	0.401	0.401	0.402	0.402	0.402	0.402
(max)	0.396	0.396	0.403	0.403	0.403	0.403	0.403	0.403	0.403	0.404	0.404
(min)	0.029	0.032	0.061	0.064	0.236	0.363	0.363	0.363	0.368	0.368	0.368
N=100 (av)	0.055	0.065	0.106	0.145	0.340	0.365	0.368	0.369	0.369	0.369	0.369
(max)	0.110	0.124	0.225	0.286	0.370	0.370	0.370	0.370	0.370	0.370	0.370
(min)	0.008	0.008	0.020	0.029	0.095	0.166	0.305	0.305	0.329	0.329	0.329
N=1000 (av)	0.008	0.009	0.023	0.033	0.116	0.191	0.316	0.317	0.329	0.329	0.329
(max)	0.009	0.010	0.026	0.037	0.137	0.216	0.328	0.328	0.328	0.328	0.328
2. EA with random operators											
Iterations	0	1	5	10	50	100	500	1000	5000	10000	20000
(min)	0.000	0.000	0.000	0.368	0.392	0.395	0.397	0.397	0.398	0.398	0.398
N=10 (av)	0.051	0.157	0.356	0.392	0.400	0.401	0.402	0.402	0.402	0.402	0.402
(max)	0.375	0.384	0.396	0.402	0.403	0.403	0.403	0.403	0.403	0.403	0.403
(min)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N=100 (av)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
(max)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
(min)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
N=1000 (av)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
(max)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

The evolutionary algorithm with heuristic operators works much better than the "blind" method also in this case. For $N = 100$ and $N = 1000$ the traditional version of EA with only "blind" operators cannot find any acceptable solution with $\mu_D > 0$. The properties of evolutionary computations displayed in that case are very similar to those shown for the stochastic case, but computations last a little bit longer. For $N = 1000$ only two simulations were conducted due to the long time of computations, but similarly like in the stochastic task this case was considered only for testing purposes. In practical applications it is easier to split the problem into shorter time periods and compute the solutions separately, beginning from the state obtained from the previous simulation.

4.4. The fuzzy system with fuzzy control

The case of fuzzy system with fuzzy controls is quite different from all the systems described before. The encoding method of the population member is quite complicated - a matrix of Nx$|\mathcal{U}|$ real values from the interval $[0, 1]$. The investigated control system was similar to the ones previously described: $|\mathcal{U}| = 32$, $|\mathcal{X}| = 20$ with fuzzy constraints imposed on states and controls and the transition matrix 32x20x20 (32 matrices 20x20). This situation is more complicated than for the deterministic control and computations, especially for $N = 1000$, last very long.

Fortunately, the best solutions (note that there is, of course, no certainty that solutions found are optimal) are found after not very many iterations, especially in comparison with the deterministic case. But the overall time of computations is about 1000 times longer than in the deterministic case, so only two test runs have been performed for $N = 1000$. The simulation results are presented in Figs. 9 and 10 and Table 4.

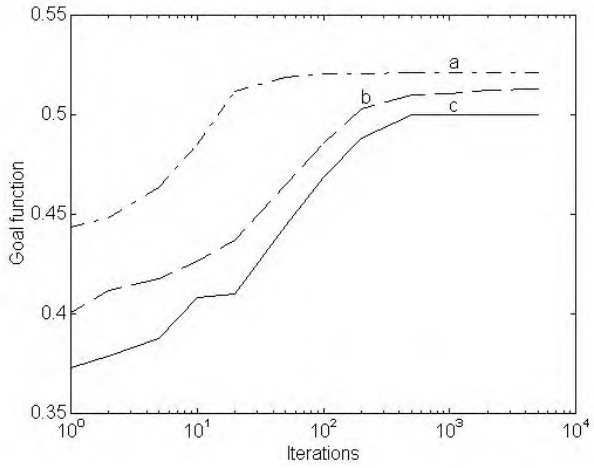


Figure 9. EA with both types of operators: a - N=10, b - N=100, c - N=1000

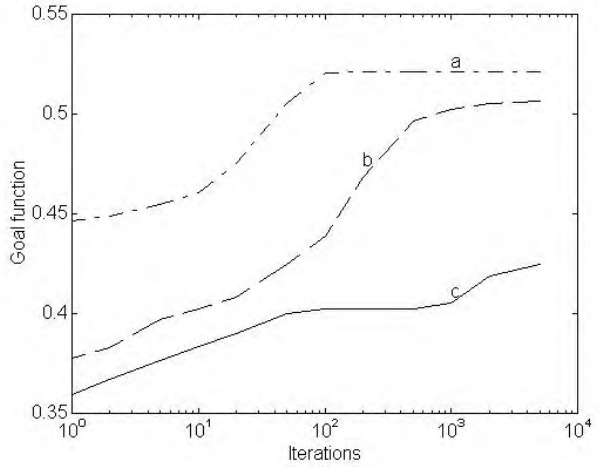


Figure 10. EA with random operators only: a - N=10, b - N=100, c - N=1000

Table 4. Numerical results of simulations for fuzzy system with fuzzy controls

1. EA with both types of operators											
Iterations	0	1	2	5	10	50	100	500	1000	2000	5000
(min)	0.419	0.421	0.425	0.431	0.449	0.498	0.516	0.521	0.521	0.521	0.521
N=10 (av)	0.439	0.443	0.449	0.464	0.485	0.519	0.520	0.521	0.521	0.521	0.521
(max)	0.474	0.474	0.476	0.508	0.521	0.521	0.521	0.521	0.521	0.521	0.521
(min)	0.364	0.375	0.377	0.391	0.399	0.399	0.448	0.502	0.502	0.506	0.506
N=100 (av)	0.378	0.400	0.412	0.418	0.427	0.465	0.486	0.510	0.511	0.513	0.513
(max)	0.408	0.428	0.454	0.454	0.454	0.486	0.512	0.516	0.516	0.516	0.516
(min)	0.341	0.345	0.358	0.375	0.400	0.440	0.463	0.500	0.500	0.500	0.500
N=1000 (av)	0.364	0.373	0.379	0.388	0.408	0.444	0.469	0.500	0.500	0.500	0.500
(max)	0.388	0.400	0.400	0.400	0.416	0.448	0.475	0.500	0.500	0.500	0.500
2. EA with random operators											
Iterations	0	1	2	5	10	50	100	500	1000	2000	5000
(min)	0.411	0.413	0.421	0.425	0.431	0.466	0.513	0.521	0.521	0.521	0.521
N=10 (av)	0.443	0.446	0.449	0.455	0.461	0.505	0.521	0.521	0.521	0.521	0.521
(max)	0.480	0.486	0.486	0.486	0.498	0.521	0.521	0.521	0.521	0.521	0.521
(min)	0.345	0.345	0.357	0.380	0.382	0.385	0.402	0.481	0.485	0.497	0.506
N=100 (av)	0.374	0.378	0.383	0.397	0.402	0.424	0.439	0.496	0.503	0.505	0.506
(max)	0.412	0.412	0.412	0.425	0.425	0.455	0.468	0.506	0.506	0.506	0.508
(min)	0.088	0.088	0.088	0.088	0.088	0.316	0.371	0.416	0.416	0.416	0.430
N=1000 (av)	0.225	0.225	0.225	0.225	0.227	0.364	0.400	0.423	0.423	0.423	0.436
(max)	0.361	0.361	0.361	0.361	0.366	0.412	0.430	0.430	0.430	0.430	0.442

In this case the difference between EA with and without heuristic operators is not so pronounced as in the previously described results, but the superiority of EA with heuristic operators is also unquestionable. This situation can be explained by the fact that in the case now considered it is quite easy to obtain any admissible solution (with $\mu_D > 0$) and even simple operators can improve such solution. The most difficult thing is to move the solution from 0. If it happens, it is possible to improve the solution using simple genetic operators, but it takes more time to acquire similar results as with heuristic operators. For deterministic controls it is very difficult to obtain any solution with $\mu_D > 0$, so only heuristic operators can deal with this problem and then efficiently improve the solution (Table 3).

5. Conclusions

Optimal multistage fuzzy control is a promising but not very often used tool for solving many difficult problems in the domain of automatics and decision making. Optimal fuzzy control is rather a difficult method, requiring heavy computations to find desired controls. Thus, the use of specialized evolutionary algorithms, described in this paper, can make it significantly easier to practically apply this method.

The computational results, presented in the previous section, show that there are computational tasks, where utilization of only random operators, similar to the widely used mutation and crossover, is useless. Only the method with specialized, knowledge-based operators, together with the simple random ones, give satisfactory results. Thus, practical application of evolutionary algorithms should involve a high level of knowledge about the problem. The more an EA knows about the solved problem, the better it works. The big advantage of evolutionary algorithms lies the fact that they can be easily developed from the completely "blind" to problem specific without changing the manner in which they work.

There are also different types of optimal multistage fuzzy control tasks (Kacprzyk, 1997 and Stańczak, 2003), for instance with infinite or fuzzy termination time, which can be solved using similar EA based methods and which will constitute the object of research in the nearest future.

References

ARABAS, J. (2001) *Wykłady z algorytmów ewolucyjnych*. Wydawnictwa Naukowo-Techniczne.

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.

Czogała, E. and Pedrycz, W. (1985) *Elementy i metody teorii zbiórw rozmytych*. PWN, Warszawa.

Esogbue, A.O., Theologidu, M. and Guo, K. (1992) On the application of fuzzy sets theory to the optimal flood control arising in water resources systems. *Fuzzy Sets and Systems* **48**, 155-172.

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, 655-660.

Kacprzyk, J. (1978) 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. (1984) Design of socio-economic regional development policies via a fuzzy decision making model. *Large Scale Systems Theory and Applications - Proceedings of Third IFAC/IFORS Symposium*. Pergamon Press, Oxford, 228-232.

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

Kacprzyk, J. (1996) Multistage control under fuzziness using genetic algorithms. *Control and Cybernetics* **25**, 1181-1215.

Kacprzyk, J. (1997) *Multistage Fuzzy Control*. John Wiley & Sons.

Kacprzyk, J. (1998) Multistage control of a stochastic system in a fuzzy environment using a genetic algorithm. *International Journal of Intelligent Systems* **13**, 1011-1023.

Kacprzyk, J.R.A., Romero, R.A. and Gomide, F.A.C. (1999) Involving objective and subjective aspects in multistage decision making and control under fuzziness: dynamic programming and neural networks. *International Journal of Intelligent Systems **14**, 79-104*.

Mulawka, J. and Stańczak, J. (1999) Genetic algorithms with adaptive probabilities of operators selection. *Proceedings of ICCIMA'99*, New Delhi, India, 464-468.

Piegat, A. (1999) *Modelowanie i sterowanie rozmyte* (*Fuzzy modelling and control*; in Polish). Akademicka Oficyna Wydawnicza EXIT, Warszawa.

Sousa, J. M. and Kaymak, U. (2001) Model predictive control using fuzzy decision functions. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics* **31**(1), 54-65.

Stańczak, J. (1999) Rozwój koncepcji i algorytmów dla samodoskonalących się systemów ewolucyjnych. (The development of the concept and algorithms for the self-improving evolutionary systems; in Polish). Ph.D. Dissertation, Warsaw University of Technology.

Stańczak, J. (2000) Algorytm ewolucyjny z populacją "inteligentnych" osobników (An evolutionary algorithm with a population of "intelligent" individuals; in Polish). *Materiały IV Krajowej Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna*, Lądek Zdrój.

STAŃCZAK, J. (2001) Evolutionary algorithm with heuristic operators in the problem of optimal fuzzy control. *Materiały V Krajowej Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna*, Jastrzębia Góra, 216-222.

STAŃCZAK, J. (2003) Biologically inspired methods for control of evolutionary algorithms. *Control and Cybernetics* **32**, 411-433.

STAŃCZAK, J. (2003) Evolutionary algorithm in the problem of optimal fuzzy control of deterministic system with indirectly given and infinite control horizon. *Materiały VI Krajowej Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna*, Łagów, 213-222.

SU, C.C. and HSU, Y.Y. (1991) Fuzzy dynamic programming: An application to unit commitment. *IEEE Transactions on Power Systems* **PS-6**, 1231-1237.

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