Raport Badawczy
Research Report

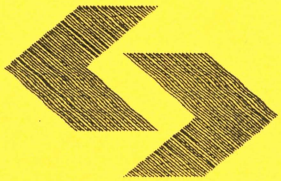
RB/44/2002

The Problem of Controlling System with Optimal Fuzzy Trajectory Using Evolutionary Algorithm

J. Stańczak

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences



POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 8373578

fax: (+48) (22) 8372772

Kierownik Pracowni zgłaszający pracę:
Prof. dr hab. inż. Janusz Kacprzyk

Warszawa 2002

The Problem of Controlling System with Optimal Fuzzy Trajectory Using Evolutionary Algorithm

Jarosław Stańczak

Instytut Badań Systemowych PAN

ul. Newelska 6

01-447 Warszawa

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 the optimal fuzzy trajectory imposed, first described by Bellman and Zadeh in [1]. An optimal control is obtained by an evolutionary algorithm, a goal trajectory and constrains imposed on the controls are given as membership functions to 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.

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

1. Introduction

Fuzzy logic is a generalization of a classic, binary logic to rules and statements, which cannot be valued as true or false. Similarly, it is used to describe situations and effects where exact values are not necessary, but quality evaluations with a very small set of possible values (i.e. small, medium, big) are enough. 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 while facts and rules are uncertain. This uncertainty may not be

caused by an inaccuracy of measurements or a too small level of knowledge about the problem but it can be its immanent property. This powerful apparatus was discovered by L. A. Zadeh, which in 1965 in the paper [12] introduced basic ideas and theoretic foundations of fuzzy sets and fuzzy logic. Since that time this domain of artificial intelligence made a great development and some more interesting data can be found in works [8], [2] and [3].

The most important practical applications of fuzzy logic are automation and control, where it is widely used from controlling trains and ships to cameras and other electronic equipment.

The multistage optimal fuzzy control is a quite different approach to fuzzy control [3] than that used in "traditional" fuzzy controllers, where control rules are fuzzy and imprecise. It deals with known and precise control algorithms and contrary to traditional method - fuzzy constrains, aims and often fuzzy models of controlled objects. Also the domain of its practical applications is rather different and covers planning, decision making and scheduling, for instance flood prevention or planning of social and economical regional development .

This paper deals with the problem of finding optimal sequence of controls (multistage control) with fixed termination time for a simple deterministic, stochastic and fuzzy objects with finite sets of states, possible controls and known, stationary transition functions. There are also fuzzy constrains imposed on controls and states of the object at every stage. The goal of the control is to maximize the resultant membership functions during that time.

Because of several limitations of conventional methods (especially for big dimensions of solved problems), usually used to obtain the optimal set of controls, a specialized evolutionary algorithm was utilized to solve this problem. The problem, evolutionary algorithm and obtained results are in details discussed later in this paper.

2. Formulation of the problem

2.1 The deterministic system

To solve the problem it is necessary to find a sequence of N 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:

$$x(t+1)=f(x(t),u(t)) \tag{2.1}$$

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 constrains imposed on the controls at each control stage t and $\mu_{C(t)}$ is a membership function to the fuzzy set of allowable controls $C(t)$. Also fuzzy goals for the following states of the system are defined by membership functions $\mu_{G(t)}$ to a set of desired states $G(t)$. Therefore the solution of the problem can be found by maximizing the expression:

$$\mu_D(\hat{u}(0),\ldots\hat{u}(N-1)|x(0)) = \max_{u(0),\ldots u(N-1)} \left[\mu_{C(0)}(u(0)) \wedge \mu_{G(1)}(x(1)) \wedge \ldots \wedge \mu_{C(N-1)}(u(N-1)) \wedge \mu_{G(N)}(x(N))\right] \tag{2.2}$$

where:

$\hat{u}(0),\ldots\hat{u}(N-1)$ - the sequence of optimal controls;

$u(0)...u(N-1)$ - controls in the following time stages;

$x(0)...x(N)$ - states of the system in the following moments;

$\mu_D(..)$ - an overall goal function for the problem;

N - the termination time;

$\mu_{G(t)}(..)$ - the membership function to fuzzy set of states ($G(t)$);

$\mu_{C(t)}(..)$ - the membership function to 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 ways: dynamic programming, branch and bound, interpolative reasoning, neural nets and also a simple genetic algorithm[2] - [3], [4], [6], but with the increasing number of control steps, their applicability become very small. The number of possible solutions in the problem, assuming fixed initial state, is equal:

$$n = |U|^N \tag{2.3}$$

where:

$|U|$ - number of possible controls;

N - termination time;

n - number of possible solutions.

Conventional methods like dynamic programming and branch and bound suffer the effects of a growth of the solved problem dimension. They are very good but for small values of N and are able to find optimal solutions. For larger ones, only heuristic, random and artificial intelligence based methods can deal with the problem effectively. Unfortunately they don't give the reliability to find 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 wait for the optimal one very long time. So it seems useful to apply the

---

[1] t-norm and its conorm - s-norm are 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 [8]. This paper deals with the simplest and the most widely used case, where $\wedge$ means minimum(...), and $\vee$ maximum(...).

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

specialized evolutionary algorithm to solve it, because it is designed specially to solve that kind of problem. That approach joins some advantages of the evolutionary algorithm with some heuristic and problem-specific methods used as "intelligent" genetic operators.

2.2 The stochastic system

In the case of the stochastic system the transition function bases 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 $|U|$ matrixes, depending on used controls). The state of the system is described by a 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.

$$P(\bar{x}(t+1)) = P(\bar{x}(t)) * P(\bar{x}(t+1) | u(t), \bar{x}(t)) \tag{2.4}$$

where:

$P(\bar{x}(t)), P(\bar{x}(t+1))$ - probability distribution on the vector of the system state in following iterations;

$P(\bar{x}(t+1) | u(t), \bar{x}(t))$ - 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 following moments.

The described system is an example of a Markov chain.

Starting from beginning state $\bar{x}(0)$ the system goes towards the terminating state $\bar{x}(N)$, according to (2.4), with fuzzy constrains imposed on states and controls at every iteration. The choice of proper control signals $u(t)$ enables to maximize the quality criterion:

$$\mu_D(\hat{u}(0),\ldots\hat{u}(N-1) | \bar{x}(0)) = \max_{u(0),\ldots u(N-1)} \left[\mu_{C(0)}(u(0)) \wedge E\mu_{G(1)}(\bar{x}(1)) \wedge \ldots \wedge \mu_{C(N-1)}(u(N-1)) \wedge E\mu_{G(N)}(\bar{x}(N))\right] \tag{2.5}$$

where:

$E\mu_{G(t)}(..)$ - an expected value of the membership function to set $G(t)$ - a fuzzy constrain imposed on state of the system, defined as:

$$E\mu_{G(t)}(\bar{x}(t)) = P(\bar{x}(t)) \cdot \mu^{T}_{G(t)}(\bar{x}(t)) = \sum_{i=1}^{i=|X|} p(x_i(t)) \cdot \mu_{G(t)}(x_i(t)) \tag{2.6}$$

$p(x_i(t))$ - a probability, that the system is in state $x_i \in X$ or in the other words, *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 $x_i \in X$;

$\mu^{T}_{G(t)}(\bar{x}(t))$ - a transpositioned vector of membership function values;

all other symbols like in (2.2) and (2.4).

Methods usually used to solve the problem are similar to that ones, described in paragraph 2.1. The size of the space of solutions can be estimated from formula (2.3). 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:

$$X(t+1) = f(X(t), U(t)) \tag{2.7}$$

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 to sets of states an controls (when necessary). The formula (2.8) shows the situation where deterministic controls are applied, the formula (2.9) describes the fuzzy system with fuzzy controls [3]:

$$\mu_{X(t+1)}(x_i(t+1)) = \max_{j \in 1..|\mathcal{X}|} \left(\mu_{X(t)}(x_j(t)) \wedge \mu_{X(t+1)}(x_i(t+1)|x_j(t), u(t)) \right) \quad (2.8)$$

$$\mu_{X(t+1)}(x_i(t+1)) = \max_{j \in 1..|\mathcal{X}|} \left(\max_{k \in 1..|\mathcal{U}|} \left(\mu_{X(t)}(x_j(t)) \wedge \mu_{X(t+1)}(x_i(t+1)|x_j(t), u_k(t)) \wedge \mu_{U(t)}(u_k(t)) \right) \right) \quad (2.9)$$

where:

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

$\mu_{X(t+1)}(x_i(t+1)|x_j(t), u(t))$ - a 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 a k-th coordinate of fuzzy control $U(t)$;

all other symbols like in the formula (2.7).

Similarly like in the case of deterministic and stochastic systems, there are fuzzy constrains imposed on states ($\mu_{G(t)}$) and controls ($\mu_{C(t)}$) of the system in every iteration. The starting 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 [3] and [5] and is called a similarity function:

$$\mu_{\overline{C}(t)}(U(t)) = 1 - d(U(t), C(t)) \text{ and } \mu_{\overline{G}(t+1)}(X(t+1)) = 1 - d(X(t+1), G(t+1)) \quad (2.10)$$

where:

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

$C(t)$ - a set of constrains imposed on controls in the iteration t;

$G(t+1)$ - a set of constrains imposed on state in the iteration $t+1$;

$\mu_{\overline{C}(t)}(U(t))$ and $\mu_{\overline{G}(t+1)}(X(t+1))$ - a measure of similarity of membership functions.

---

[3] In the considered case a measure of distance in Hamming sense was applied in computer simulations.

A quality criterion for the system with fuzzy control shows the formula (2.11), with deterministic control the formula (2.12):

$$\mu_D(\hat{U}(0),\ldots\hat{U}(N-1)\mid X(0)) = \max_{U(0),\ldots U(N-1)}\left[\mu_{\bar{C}(0)}(U(0)) \wedge \mu_{\bar{G}(1)}(X(1)) \wedge \ldots \wedge \mu_{\bar{C}(N-1)}(U(N-1)) \wedge \mu_{\bar{G}(N)}(X(N))\right] \tag{2.11}$$

$$\mu_D(\hat{u}(0),\ldots\hat{u}(N-1)\mid X(0)) = \max_{U(0),\ldots U(N-1)}\left[\mu_{C(0)}(u(0)) \wedge \mu_{\bar{G}(1)}(X(1)) \wedge \ldots \wedge \mu_{C(N-1)}(u(N-1)) \wedge \mu_{\bar{G}(N)}(X(N))\right] \tag{2.12}$$

where:

$\hat{U}(0),\ldots\hat{U}(N-1)$ - a sequence of optimal fuzzy controls;

$U(0)...U(N-1)$ - fuzzy controls in following moments;

$\hat{u}(0),\ldots\hat{u}(N-1)$ - the sequence of optimal deterministic controls;

$u(0)...u(N-1)$ - deterministic controls in following time stages;

$X(0),...X(N)$ - states of the system in following iterations;

$\mu_D(..)$ - an overall goal function for the problem;

N - a termination time;

$\wedge$ - a t-norm sign;

all other symbols like in the formula (2.10).

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 (2.3), for fuzzy controls the solution space is continuous and all possible solutions are in a hypercube $\langle 0, 1\rangle^{N*|U|}$. A high number of dimensions in the problem make it difficult to solve and the evolutionary algorithm is a good choice to solve that problem.

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

In spite of their universality, the evolutionary algorithm should be properly adjusted to solve a raised problem. Unchanged remains only the core idea of its work: thanks 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 coding[4] of solutions and an invention of proper genetic operators for that problem. Genetic evaluations stop after fixed number of generations, when satisfying results are obtained or there are no changes of fitness function during some number of generations.

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

Fig. 3.1. The evolutionary algorithm.

3.1 The individual encoding for AE

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 an actual solution is stored in a sequence of controls [3] (assuming a fixed start state). So the genotype of every member of the population is a sequence of indexes (integer values) of control signals from the set U for following steps of the controlled object operation. This method of the solution encoding is

---

[4]Applying a binary coding, which was not long ago treated as a base of evolutionary algorithms, is not very useful for this problem and now is used only when it is easy and helpful to apply it.

very effective and gives the advantage of elimination of not allowed solutions. A real member of the population, used for evolutionary computations, contains an integer table of N values of controls, 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. 3.2 presents a scheme of an individual of the population with deterministic controls.

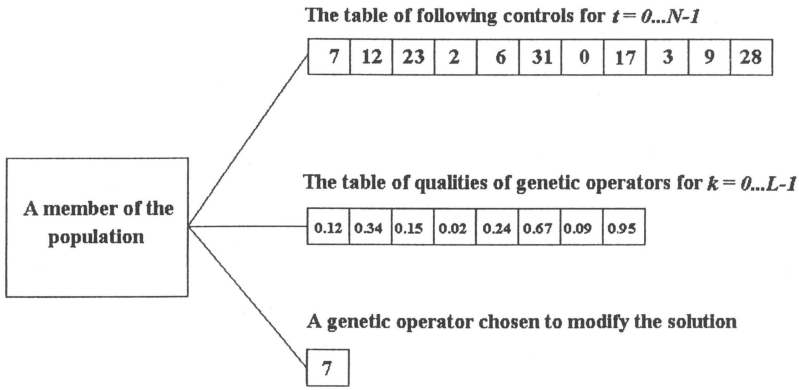


Fig. 3.2. A member of the population of solutions for the case of deterministic controls.

3.1.2 The case of fuzzy controls

For the fuzzy system, described by equation (2.7) and (2.8), fuzzy controls are applied. The fuzzy controls are vectors of real values. These real numbers are values of the membership function to the set of allowable controls $\mathcal{U}$. Different values of these real numbers (different membership functions) make different control signals. It implies that a member of the population contains N (number of control steps) vectors of membership function values, so it is a matrix $N \times |\mathcal{U}|$ of real values from the interval $\langle 0,1 \rangle$. This method of encoding also provides the situation that only allowable solutions may appear.

Similarly like in the case of deterministic controls a member of the population contains also a number of chosen genetic operator and a vector of qualities of operators. The method of operator selection is precisely described in paragraph 3.4. Fig. 3.3 presents a scheme of an individual of the population with fuzzy controls.

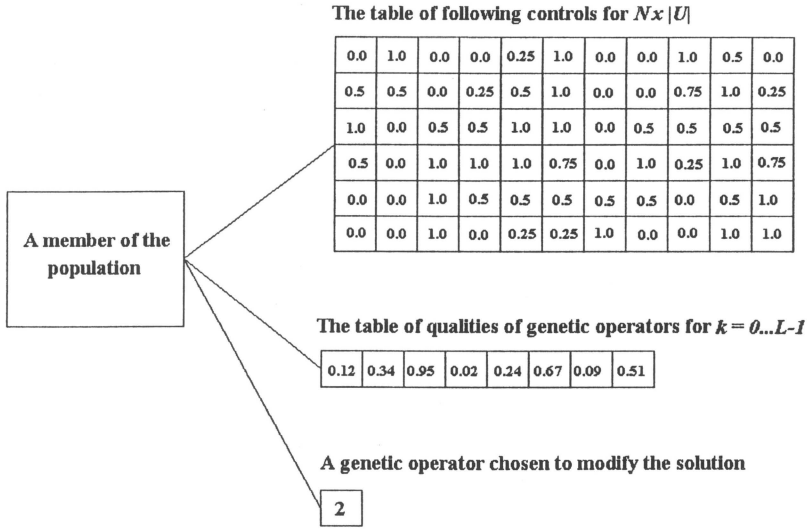


Fig. 3.3. A member of the population of solutions for the case of fuzzy controls.

3.2 The fitness function

In all considered cases, the quality functions of corresponding problems (2.2, 2.5, 2.11 or 2.12) have appropriate properties to become directly fitness functions for evolutionary algorithm:

- non-negative values of membership functions are from the interval $\langle 0,1 \rangle$;
- the better quality of the individual, the higher value of the problem's quality function (maximization);

- fuzzy constrains 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.

So problem's quality functions are used as fitness functions for EA in the 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 think out for both methods of encoding (with integer and real numbers) and similar to the widely used: mutation, inversion and crossover. Also multiple versions of described operators were applied. Simulations of evolutionary algorithm with such "blind" operators proved that some more sophisticated, enriched with some heuristics should be used. Random operators have big difficulties to find even very poor but non-zero solutions, especially for problems with deterministic control and $N>10$. Two methods were tested to overcome that problem: a non-random generation of the starting population with random operators [11] and the randomly generated beginning population with heuristic and "blind" operators. The second solution demonstrated to act much better - details are presented in the next paragraph.

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. Applying rather big number of genetic operators (about 12) caused some problems with choosing appropriate probabilities of operators appearance. The method, which is a good solution for that situation is presented in paragraph 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 coding strings;

- mutation - a random change of one randomly selected control to another from the set U;
- inversion - an inversion of a randomly chosen subset of the solution.

Beside it the same operators appeared in multiple version - 3 repetitions of the operator. These operators belong to the class of "blind", random operators, which are specialized to used data structures as members of the population of solutions, but do not know anything about the solved problem. As it was previously mentioned, experiments with such a set of operators gave poor results [11], especially for $N>10$.

Three heuristic operators for deterministic control are invented:

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

The first of heuristic operators resembles mutation but with some heuristics added, so is called "intelligent" mutation. Traditional mutations works like operator *not* on randomly chosen bit of the solution. In the case of the solved problem, mutation is more complicated, and in a randomly chosen place of the coding string, changes one control to another randomly chosen from the set U. "Intelligent" mutation acts almost the same, but new control is randomly chosen from a set of allowable controls[5] (i.e. with $\mu_{C(t)}(t)>0$) for actual step t. This way of generating new solutions is potentially more effective than in the case of 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 $t=N-1$ and tries to correct places, where local membership function drops to 0 or the lowest value

---

[5] Sets of allowable controls for every iteration are created during the initialization of the EA, and are also used for generating of not random starting population and in other heuristic operators.

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 to find betters. If it succeeds, a list of possible better solutions is made. Then one of the controls from the list is chosen[6] and replaces formerly used one. If the list for some value of t is empty (it may happen in some circumstances) no operation is performed and operator ends its work. The algorithm of the operator shows fig. 3.4. All symbols on fig. 3.4 and fig. 3.5 have the same meaning as in paragraph 2.1 and presented algorithms concern the case of deterministic system under control, but they can easily be extended (which was done in the computer simulations) to all cases with deterministic control.

1. $i: =1$;
2. Evaluation $\mu(i)$ (a local membership function value)

$$\mu(i): =\mu_{C(i-1)}(u(i-1))\wedge\mu_{G(i)}(x(i));$$

3. If $\mu(i) \leq \mu_D(u(0),...,u(N-1)|x(0))$

   a) creation a list of allowable pairs $(u(i-1), x(i))$, for which

$$\mu(i) >\mu_D(u(0),...,u(N-1)|x(0));$$

   b) selection of one pair from the list (there are possible several methods of choosing);

   c) modification of the solution on the position i (if the list of possible pairs is empty, the position i is unchanged and operator stops);

4. $i: =i+1$;
5. If $i\leq N$ then go to 2;
6. Stop.

Fig. 3.4. The algorithm of the "forward" genetic operator.

---

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

The last heuristic operator is called "backward", because it works similarly to the first one but in the reverse order (from $t = N-1$ to $t = 0$). In the iteration (control) when a low value of membership function is found, an appropriate control is chosen (like in the case of "forward" operator) from the list of allowable controls and states, produced by an operator (only for that iteration). The controls are chosen in the manner, that preserves the sequence of earlier established states of the controlled object.

1. $i: =N$;
2. Evaluation $\mu(i)$ (a local membership function value)

$$\mu(i): =\mu_{C(i-1)}(u(i-1))\wedge\mu_{G(i)}(x(i))$$

for all possible pairs $(u(i-1), x(t))$;

3. Selection of one pair so as:

$$\mu(i)>\mu_D(u(0),...,u(N-1)|x(0))$$

4. Selection from the transition matrix a state $x(i-1)$, for which:

$$x(i): =f(x(i-1),u(i-1));$$

5. Modification of i-th position of the solution with chosen control number (if no control is selected or state x(i-1) cannot be found, solution on position i remains unchanged and operator stops);
6. $i: =i-1$;
7. If $i>0$ then go to 2;
8. Stop.

Fig. 3.5. The algorithm of the "backward" genetic operator.

Similarly like in the "forward" operator, the list of possible better solutions may be empty and then the operator stops for $t>0$[7], but changes performed before that point are valid. In that case

---

sequences of controls.

[7] 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 if $x(1)$ is equal $f(x(0),u(0))$ or not and no other moves are allowed.

the operator rather does not improve the solution but works like some kind of random genetic operator. The detailed algorithm of the method is presented on fig. 3.5.

The "backward" genetic operator is successfully applied only in the case of the deterministic system, because it is easy to find an inverse transition function. In the case of the stochastic system an inverse matrix of probabilities is required to perform the operator, but an inverse simulation of the system operation gives states with negative or greater than *1* "probabilities" in a state vector. Numerical errors and the fact that inverse matrixes contain also negative or greater than 1 values cause it. So in the case of the stochastic system its usefulness is rather problematic and the operator works more like a kind of a random one. For the fuzzy system it is impossible to perform the inverse simulation of the system operation and the "backward" operator can't work.

3.3.2 Genetic operators for fuzzy control

A real value matrix encoding requires using 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 $\langle 0,1 \rangle$;
- 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 little resembles a manner the "forward" heuristic operator works. The operator follows the sequence of control vectors from $t = 0$ to $t = N-1$, finds places where the membership function drops to the lowest values (the lowest value is a value of a particular solution goal function) and tries to increase them by changing values of membership functions in the control vector. The values are changed to make vectors of fuzzy states and controls closer to fuzzy constrains imposed on the system in the current time step. The algorithm of the heuristic operator for fuzzy control is shown on fig. 3.6 (all symbols like in formulae 2.7, 2.8 and 2.11).

1. $i:=1$;
2. Evaluation $\mu(i)$ (a local membership function value):

$$\mu(i) = \mu_{\overline{C}(i-1)}(U(i-1)) \wedge \mu_{\overline{G}(i)}(X(i))$$

3. If $\mu(i) \leq \mu_D(U(0),...,U(N-1)|X(0))$ then:

   a) if $\mu_{\overline{C}(i-1)}(U(i-1)) \leq \mu_D(U(0),\ldots,U(N-1)|X(0))$ then:

   - $U_p(i-1)=C(i-1)$;
   - $\mu_p(i) = \mu_{\overline{C}(i-1)}(U_p(i-1)) \wedge \mu_{\overline{G}(i)}(X_p(i))$;
   - if $\mu_p(i)>\mu(i)$ then $U(i-1)=U_p(i-1)$, $X(i)=X_p(i)$ else go to 6;

   b) if $\mu_{\overline{G}(i)}(X(i)) \leq \mu_D(U(0),\ldots,U(N-1)|X(0))$ then:

   - $U_p(i-1)=(|x|*C(i-1)+|u|*G(i))/(|x|+|u|)$;
   - $\mu_p(i) = \mu_{\overline{C}(i-1)}(U_p(i-1)) \wedge \mu_{\overline{G}(i)}(X_p(i))$;
   - if $\mu_p(i)>\mu(i)$ then $U(i-1)=U_p(i-1)$, $X(i)=X_p(i)$ else go to 6;

4. $i: =i+1$;
5. If $i \leq N$ then go to 2;
6. Stop.

Fig. 3.6. The algorithm of the heuristic operator for problem with fuzzy control.

3.4 EA with ranking of genetic operators

Using such 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 works [7], [9], [10]) it is assumed that an operator that generates good results should have bigger probability and more frequently effect the population. But it is very likely that the operator, that is good for one individual, gives worse effects for another, for instance because of its location in the domain of possible solutions. So every individual may have its own preferences. Every individual has a vector of floating point numbers - q (beside encoded solution). Each number corresponds to one genetic operation. It is a measure of quality of the genetic operator. The higher the number is, the higher is the probability of the operator. This relationship may be written as follows:

$$p_{ij}(t) = \frac{q_{ij}(t)}{\sum_{i=1}^{i=L(t)} q_{ij}(t)} \tag{3.1}$$

where:

$q_{ij}(t)$ - a quality coefficient of the i-th operation in the moment t for j-th member of population;

$p_{ij}(t)$ - a probability of an appearance of the i-th operation in the moment t for j-th member of population;

$L(t)$ - a number of genetic operators (may vary during genetic computations).

This ranking becomes a base to compute 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 gathered experience one can maximize chances of its offspring to survive. The quality factors are computed according to the formula (3.2):

$$q_{ij}(t+1) = \begin{cases} q_{0ij}(t) + x_{ij}(t+1) + \alpha_{ij}(t) * q_{lj}(t) \text{ for } i = l \\ q_{ij}(t) \qquad\qquad\qquad\qquad \text{for other } i \end{cases} \tag{3.2}$$

where:

$q_{ij}(t+1)$, $q_{ij}(t)$ - a quality of *i-th* operation for *j-th* individual in following generations;

$q_{0ij}(t)$ - a credit value, which can be modified during iterations;

$x_{ji}(t)$[8] - an improvement of the problem's quality function, obtained by *i-th* operation for *j-th* member of population. In the case of lack of the improvement equal to zero, $x_{ji}(t)=Q_j(t) - Q(t-1)$ (maximization), $Q_j(t)$ - solution of *j-th* individual, $Q(t-1)$ - the global best solution, found till the moment $t-1$;

$\alpha(t)$ - a coefficient of forgetting, $\alpha \in (0, 1)$, its value also can be adjusted during evolutionary process[9];

l - an index of the operator chosen for modification of the particular solution.

The first element of the formula (3.2) - $q_0(t)$ plays a role of credit - a small value, which supports small level of q_{ij} even if the operator does not give any advantages for a long term. Dropping this value to zero would eliminate corresponding to it operation for current individual and for its possible offspring. This fact is not profitable, because it is possible that they 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 (as it can be noticed later from the presented simulation results).

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

---

[8] In the work [9] the improvement of the problem's goal function is normalized to value between 0 and 1, but in the solved problem it is unnecessary, because values of membership functions belong to this interval.

[9] The case with adjusted values of $\alpha_{ij}(t)$ and $q_{0ij}(t)$ is shown in the paper [10].

The third part of the formula (3.2) remembers old achievements of an operator multiplied by forgetting factor $\alpha(t)$. It is responsible for balancing influence the quality factor of old and new improvements of the operator. Decreasing the value worked out by an operator is introduced to make the evolutionary algorithms more flexible. It should be noticed that some genetic operators may achieve good results in some phase of simulation, and then draw out their abilities, but others, probably better in next phases, would have small probabilities of appearance, so it would take a lot of time to change this situation. The effect of forgetting former achievement can overcome this problem. When operators don't change the global best solution for some time, the probabilities of operators become equal. After every generation only bounded with the chosen operator value $q_{ij}(t+1)$ is updated, the others remain unchanged. Only one operator is executed in one generation for one individual, so there is no reason to change coordinates corresponding to the other, not selected operators.

3 5. Controlled selection

The applied selection method consists of two methods with different properties: a histogram selection (increases the diversity of the population) and a deterministic roulette (strongly promotes best individuals) [9], which are selected in random during the execution of the algorithm. The probability of executing of the selection method is obtained from the formula 3. 3.

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

$$p_{his}(t+1)=\begin{cases} p_{his}(t)\cdot(1-a) \quad for\ \max(F_{mean}-F_{\min},F_{\max}-F_{mean})>3\cdot\sigma(F) \\ p_{his}(t)\cdot(1-a)+a \quad for\ \max(F_{mean}-F_{\min},F_{\max}-F_{mean})<0.5\cdot\sigma(F) \\ p_{his}-(0.5-p_{det}(t))\cdot a\ for \begin{pmatrix} \max(F_{mean}-F_{\min},F_{\max}-F_{mean})\geq 0.5\cdot\sigma(F)\ and \\ \max(F_{mean}-F_{\min},F_{\max}-F_{mean})\leq 3\cdot\sigma(F) \end{pmatrix} \end{cases} \tag{3.3}$$

where:

p_{his} - probability of histogram selection appearance (1-p_{his} is a probability of deterministic roulette method p_{det});

F_{mean}, F_{min}, F_{max} - average, minimal and maximal values of fitness function in the population;

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

a - a small value to change probability p_{his}, in simulations $a=0.05$.

4. Results of simulations.

Results, obtained using the simple genetic algorithm [3], [5], [6], were promising, but not accurate, especially for bigger number of control steps. To overcome this problem it is necessary to use the specialized evolutionary algorithm, which was described in the previous paragraph. Results of genetic evaluations are presented in this paragraph. Common parameters of conducted computer simulations are listed below:

- presented results are minimal/mean/maximal values obtained from 100 simulations for each problem, in cases for which a smaller value of simulations were conducted (due to long time of the simulation) an exact value of them is presented;
- all figures present only mean values;
- mean 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. A half of the population is obtained randomly from all possible controls. The second part is generated randomly but only from controls which

have a value of their membership functions for current step greater than *0* (it gives better staring points than simple random method);

- for cases with and without heuristic operators the same problem was solved;
- all solved problems were generated randomly;
- strategy $(\mu+\lambda)$[10] was applied, where $\mu=60$, $\lambda=320$;
- for every simulation the same methods of individual selection and ranking of genetic operators were used.

4.1 The deterministic system

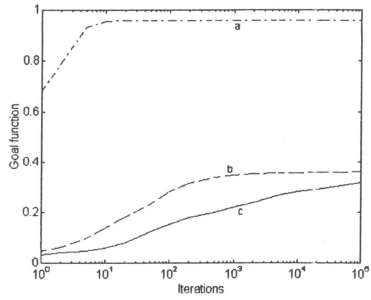
In the conducted simulations three sizes of problems were considered, for $N=10$, 100 and 1000. For all cases $|U|=32$, $|X|=20$, with appropriate number of fuzzy constrains for controls and stages for all sizes of the problem. It would be difficult to present all parameters of solved problems in this paper, so only a short description of them is presented here. The transition function is a 20×32 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 $\langle 0, 1 \rangle$, but it is chosen only for simplicity and every problem can be scaled to that interval. The problem's goal function (2.2) is also a fitness function of the evolutionary algorithm. Membership functions for imposed constrains are trapezoid functions on X and U. For every element respectively X and U a membership function value is given. Those values put together from the smallest to the highest value of the argument form a vector, which is a fuzzy constrain $G(t)$ or $C(t)$.

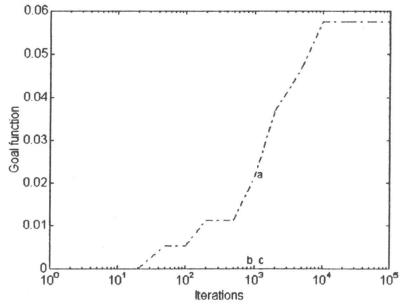
Fig 4.1 presents graphics comparisons of obtained results (mean values of best results from 100 simulations). Tab 4.1 contains numerical values of presented on fig 4.1 results and

---

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

additionally values of the worst and the best solutions obtained at presented time moments during 100 simulations. The comparison of presented minimal, average and maximal values during genetic computations may be helpful to realize that obtained results can be very different in simulations conducted with the same parameters.





1) EA with both types of operators;

2) EA with random operators only;

Fig 4.1 Graphic presentation of acquired results, (for all figures: a - N =10, b - N =100, c - N =1000).

1. EA with both types of operators												
Iterations	**0**	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**5000**	**10000**	**50000**	**100000**
(min)	0.0000	0.1652	0.6667	0.8911	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586
N=10 (av)	0.0051	0.6851	0.9314	0.9572	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586
(max)	0.1652	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586	0.9586
(min)	0.0000	0.0000	0.0167	0.0464	0.1659	0.1659	0.3027	0.3033	0.3231	0.3374	0.3374	0.3374
N=100 (av)	0.0000	0.0462	0.0959	0.1382	0.2336	0.2825	0.3384	0.3470	0.3560	0.3571	0.3590	0.3624
(max)	0.0000	0.1769	0.2585	0.3027	0.3374	0.3764	0.3764	0.3764	0.3764	0.3764	0.3764	0.3764
(min)	0.0000	0.0043	0.0207	0.0268	0.0883	0.1196	0.1655	0.1825	0.2273	0.2594	0.2794	0.2823
N=1000 (av)	0.0000	0.0321	0.0481	0.0584	0.1227	0.1529	0.1995	0.2193	0.2691	0.2826	0.3057	0.3189
(max)	0.0000	0.3418	0.3471	0.3471	0.3471	0.3471	0.3471	0.3471	0.3471	0.3534	0.3883	0.3899
2. EA with random operators												
Iterations	**0**	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**5000**	**10000**	**50000**	**100000**
(min)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
N=10 (av)	0.0000	0.0000	0.0000	0.0000	0.0054	0.0054	0.0113	0.0214	0.0470	0.0575	0.0575	0.0575
(max)	0.0000	0.0000	0.0000	0.0000	0.1345	0.2490	0.6532	0.8654	0.9586	0.9586	0.9586	0.9586
(min)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
N=100 (av)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(max)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(min)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
N=1000 (av)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(max)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Tab 4.1 Numerical results of simulations for deterministic system.

To be exact, 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 give the method with both types of operators, especially for higher values of N (*100, 1000*). The method, with random operators and randomly generated starting population, gives very poor results. Poor results for that method are connected with a very little probability of random generating any allowable solution, as it can be seen, it happened several times only for $N=10$.

4.2 The stochastic system

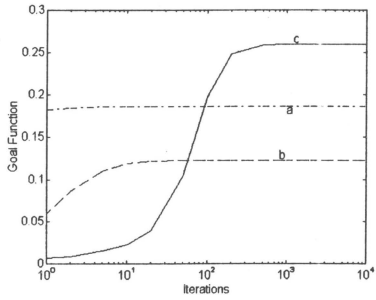
In this section, results obtained for the stochastic system are discussed. Evolutionary computations were conducted for $N=10$, $N=100$ and $N=1000$ (for $N=1000$ computations last very long so only a few simulations were performed). Similarly to the previously described case, the system with *20* states and *32* possible controls with proper number of fuzzy constrains (constructed in the same manner) for them was assumed. The conditional transition matrix for that case contains probabilities of transitions among states, considering possible controls. Precisely, it is a set of *32* matrixes *20 x 20*, depending on applied controls.

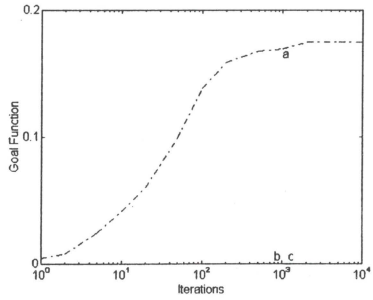
Fig. 4.2 presents results graphically and Tab. 4.2 shows numerical values obtained during computations.

Results presented in this section also prove the thesis that heuristic operators are very powerful and can significantly speed up evolutionary computations. Especially for larger values of N. It should be noticed that stochastic systems need less iterations that deterministic ones to find good solutions, but one iteration lasts much longer and the overall time of evolutionary computations is much longer than in the deterministic case.

1. EA with both types of operators										
Iterations	**0**	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**5000**	**10000**
(min)	0.1480	0.1632	0.1843	0.1858	0.1859	0.1859	0.1859	0.1863	0.1863	0.1863
N=10 (av)	0.1726	0.1823	0.1858	0.1861	0.1862	0.1862	0.1862	0.1863	0.1863	0.1863
(max)	0.1854	0.1862	0.1862	0.1863	0.1863	0.1863	0.1863	0.1863	0.1863	0.1863
(min)	0.0159	0.0210	0.818	0.0903	0.1216	0.1218	0.1225	0.1225	0.1225	0.1225
N=100 (av)	0.0356	0.0599	0.1104	0.1188	0.1224	0.1225	0.1225	0.1225	0.1225	0.1225
(max)	0.0882	0.1178	0.1225	0.1225	0.1225	0.1225	0.1225	0.1225	0.1225	0.1225
(min)	0.0044	0.0049	0.0092	0.0185	0.0691	0.1697	0.2583	0.2594	0.2594	0.2594
N=1000 (av)[11]	0.0060	0.0068	0.0159	0.0228	0.1038	0.1979	0.2590	0.2594	0.2594	0.2594
(max)	0.0115	0.0115	0.0260	0.0369	0.1299	0.2333	0.2593	0.2594	0.2594	0.2594
2. EA with random operators										
Iterations	**0**	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**5000**	**10000**
(min)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1749	0.1749
N=10 (av)	0.0000	0.0046	0.0244	0.0415	0.0996	0.1389	0.1678	0.1696	0.1749	0.1749
(max)	0.0000	0.1727	0.1747	0.1749	0.1749	0.1749	0.1749	0.1749	0.1749	0.1749
(min)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
N=100 (av)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(min)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(min)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
N=1000 (av)[10]	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(min)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Tab 4.2 Numerical results of simulations for stochastic system.





1) EA with both types of operators;

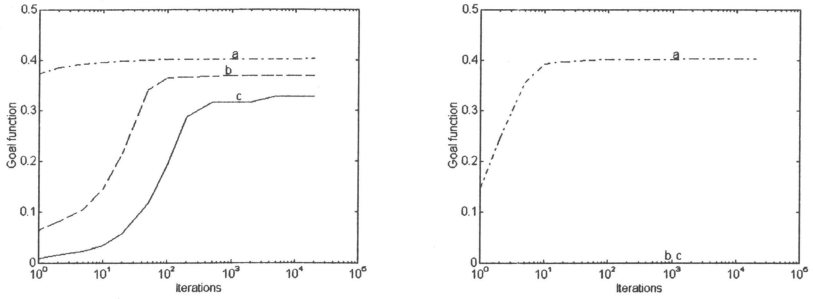
2) EA with random operators only;

Fig 4.2 Graphic presentation of acquired results (for both figures: a - N =10, b - N =100, c - N = 1000).

[11] Results obtained from 15 simulations.

4.3 The fuzzy system with deterministic control

AE to solve the problem of optimal fuzzy control for fuzzy system with deterministic controls was tested using object described by a transition function with *20* fuzzy states and *32* deterministic controls (*32* matrixes *20 x 20*) for *N=10*, *N=100* and *N=1000* with fuzzy constrains 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, that consists of *20* membership values to elements of $\mathcal{X}$. Results of simulations are presented on Fig 4.3 and Tab. 4.3.



1) EA with both types of operators; 2) EA with random operators only;

Fig 4.3 Graphic presentation of acquired results (for both figures: a - N =10, b - N =100, c - N= 1000).

The evolutionary algorithm with heuristic operators acts much better than classic one also in this case. For *N=100* and *N=1000* traditional version of AE with only "blind" operators can't find any acceptable solution with $\mu_D>0$. Properties of evolutionary computations in that case are very similar to the stochastic case, but computations last a little longer than in the stochastic case and for *N=1000* only several simulations were conducted. It should be also noticed that differences between minimal, average and maximal values are very small in cases of stochastic and fuzzy control, for the deterministic case are significantly bigger

1. EA with both types of operators												
Iterations	**0**	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2000**	**5000**	**10000**	**20000**
(min)	0.2586	0.3409	0.3820	0.3900	0.3934	0.3934	0.3969	0.3977	0.3977	0.3977	0.3977	0.3977
N=10 (av)	0.3577	0.3736	0.3930	0.3956	0.4003	0.4009	0.4015	0.4019	0.4020	0.4022	0.4022	0.4022
(max)	0.3957	0.3965	0.4027	0.4027	0.4027	0.4027	0.4027	0.4027	0.4027	0.4027	0.4042	0.4042
(min)	0.0287	0.0318	0.0615	0.0640	0.2363	0.3633	0.3635	0.3635	0.3681	0.3681	0.3681	0.3681
N=100 (av)	0.0555	0.0652	0.1063	0.1454	0.3405	0.3651	0.3684	0.3686	0.3688	0.3689	0.3691	0.3694
(max)	0.1103	0.1242	0.2253	0.2858	0.3702	0.3702	0.3702	0.3702	0.3702	0.3702	0.3702	0.3702
(min)	0.0078	0.0078	0.0202	0.0291	0.0952	0.1665	0.3046	0.3046	0.3046	0.3287	0.3287	0.3287
N=1000 (av)[12]	0.0084	0.0090	0.0229	0.3331	0.1164	0.1911	0.3165	0.3167	0.3167	0.3287	0.3287	0.3287
(max)	0.090	0.0102	0.0256	0.0375	0.1375	0.2157	0.3285	0.3287	0.3287	0.3287	0.3287	0.3287
2. EA with random operators												
Iterations	**0**	**1**	**5**	**10**	**50**	**100**	**500**	**1000**	**2000**	**5000**	**10000**	**20000**
(min)	0.0000	0.0000	0.0000	0.3684	0.3919	0.3954	0.3966	0.3969	0.3977	0.3977	0.3977	0.3977
N=10 (av)	0.0508	0.1575	0.3559	0.3919	0.3997	0.4008	0.4019	0.4021	0.4023	0.4023	0.4024	0.4024
(max)	0.3750	0.3840	0.3957	0.4024	0.4027	0.4027	0.4027	0.4027	0.4027	0.4027	0.4027	0.4027
(min)	0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
N=100 (av)	0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(max)	0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(min)	0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
N=1000 (av)[11]	0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
(max)	0.0000	0.000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Tab. 4.3 Numerical results of simulations for fuzzy system with deterministic controls.

4.4 The fuzzy system with fuzzy control

The case of fuzzy system with fuzzy controls is quite different from all systems described above. The encoding method of the population member is quite complicated - a matrix of $Nx|U|$ real values from interval $\langle 0,1 \rangle$. The investigated control system was similar to previously described: $|U|=32$, $|X|=20$ with fuzzy constrains imposed on states and controls and the transition matrix $32 \times 20 \times 20$ (32 matrixes 20×20). This system is quite complicated and computations, especially for $N=1000$, last very long. Fortunately best solutions[13] 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 a few test have been performed for $N=1000$.

The simulation results are presented on Fig. 4.4 and Tab. 4.4.

---

[12] Results obtained from 2 simulations.

[13] Unluckily for AE there is no certainty that found solutions are optimal.

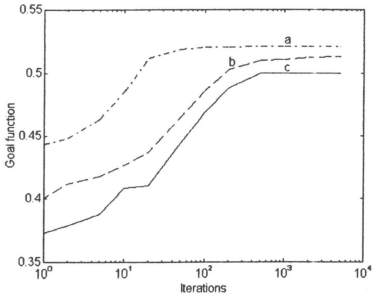
In this case the difference between AE with and without heuristic operators is not so high as in previously described results, but the superiority of AE with heuristic operators is also unquestionable. This situation can be explained by the fact that in that case it is quite easy to obtain any allowable 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 happen, 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 (Tab. 4.3).

1. EA with both types of operators											
Iterations	**0**	**1**	**2**	**5**	**10**	**50**	**100**	**500**	**1000**	**2000**	**5000**
(min)	0.4195	0.4207	0.4252	0.4308	0.4493	0.4979	0.5159	0.5206	0.5210	0.5212	0.5213
N=10 (av)	0.4386	0.4432	0.4487	0.4637	0.4852	0.5189	0.5205	0.5213	0.5213	0.5213	0.5213
(max)	0.4740	0.4740	0.4760	0.5081	0.5210	0.5213	0.5213	0.5213	0.5213	0.5213	0.5213
(min)	0.3637	0.3755	0.3775	0.3915	0.3994	0.3994	0.4478	0.5018	0.5018	0.5060	0.5060
N=100 (av)	0.3781	0.4005	0.4120	0.4178	0.4266	0.4646	0.4856	0.5103	0.5108	0.5130	0.5132
(max)	0.4082	0.4283	0.4537	0.4537	0.4537	0.4859	0.5123	0.5160	0.5160	0.5160	0.5160
(min)	0.3406	0.3454	0.3578	0.3748	0.4004	0.4397	0.4629	0.5000	0.5000	0.5000	0.5000
N=1000 (av)[14]	0.3645	0.3729	0.3791	0.3876	0.4082	0.4439	0.4688	0.5000	0.5000	0.5000	0.5000
(max)	0.3885	0.4004	0.4004	0.4004	0.4160	0.4480	0.4747	0.5000	0.5000	0.5000	0.5000
2. EA with random operators											
Iterations	**0**	**1**	**2**	**5**	**10**	**50**	**100**	**500**	**1000**	**2000**	**5000**
(min)	0.4114	0.4129	0.4208	0.4255	0.4309	0.4664	0.5134	0.5209	0.5210	0.5211	0.5213
N=10 (av)	0.4431	0.4465	0.4491	0.4546	0.4607	0.5055	0.5206	0.5212	0.5213	0.5213	0.5213
(max)	0.4804	0.4862	0.4862	0.4862	0.4977	0.5211	0.5213	0.5213	0.5213	0.5213	0.5213
(min)	0.3447	0.3447	0.3567	0.3798	0.3822	0.3846	0.4017	0.4810	0.4847	0.4968	0.5057
N=100 (av)	0.3743	0.3776	0.3832	0.3973	0.4024	0.4245	0.4387	0.4965	0.5026	0.5054	0.5064
(max)	0.4121	0.4121	0.4121	0.4247	0.4247	0.4550	0.4679	0.5063	0.5065	0.5065	0.5081
N=1000 (av)[15]	0.3583	0.3591	0.3672	0.3763	0.3833	0.3999	0.4026	0.4026	0.4053	0.4191	0.4247

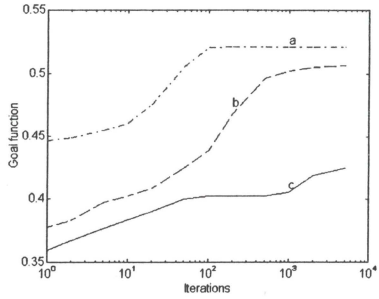
Tab 4.4 Numerical results of simulations for fuzzy system with fuzzy controls.

[14] Two simulations have been performed (32 days of computations).

[15] Only one simulation has been performed (25 days of computations).



1) EA with both types of operators;



2) EA with random operators only;

Fig 4.4 Graphic presentation of acquired results (for both figures: a - N =10, b - N =100, c - N=1000).

5. Conclusions

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

References

[1] R. E. Bellman, L. A. Zadeh: "Decision making in a fuzzy environment", Management Science, 17, pp. 141-164;

[2] E. Czogała, W. Pedrycz: "Elementy i metody teorii zbiorów rozmytych", Warszawa PWN 1985.

[3] J. Kacprzyk: "Multistage Fuzzy Control", John Wiley & Sons, 1997.

[4] J. Kacprzyk: "Multistage Control of a Stochastic System in a Fuzzy Environment Using a Genetic Algorithm", International Journal of Intelligent Systems, Vol. 13, 1998, pp. 1011-1023;

[5] J. Kacprzyk: "Multistage control under fuzziness using genetic algorithms", Control and Cybernetics, Vol. 25, 1996, pp. 1181-1215;

[6] J. Kacprzyk, R. A. Romero, F. A. C. Gomide: "Involving Objective and Subjective Aspects in Multistage Decision Making and Control under Fuzziness: Dynamic Programming and Neural Networks", International Journal of Intelligent Systems, Vol. 14, 1999, pp. 79-104;

[7] J. Mulawka, J. Stańczak: "Genetic Algorithms with Adaptive Probabilities of Operators Selection", Proceedings of ICCIMA'99, New Delhi, India, 1999.

[8] A. Piegat: "Modelowanie i sterowanie rozmyte", Akademicka Oficyna Wydawnicza EXIT, Warszawa, 1999.

[9] J. Stańczak: "Rozwój koncepcji i algorytmów dla samodoskonalących się systemów ewolucyjnych", Ph.D. Dissertation, Politechnika Warszawska, 1999.

[10] J. Stańczak: "Algorytm ewolucyjny z populacją "inteligentnych" osobników", Materiały IV Krajowej Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna, Lądek Zdrój, 2000;

[11] J. Stańczak: "Evolutionary Algorithm with Heuristic Operators in the Problem of Optimal Fuzzy Control", Materiały V Krajowej Konferencji Algorytmy Ewolucyjne i Optymalizacja Globalna, Jastrzębia Góra, 2001;

[12] L. A. Zadeh: "Fuzzy sets", Information and Control, vol. 8, 1965, pp. 338-353.