85/2007

Raport Badawczy

Research Report

RB/36/2007

Design of problem specific evolutionary operators

J. Stańczak, Z. Nahorski

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) 3810100

fax: (+48) (22) 3810105

Kierownik Pracowni zgłaszający pracę:
prof. dr hab. inż. Zbigniew Nahorski

Warszawa 2007

DESIGN OF PROBLEM SPECIFIC EVOLUTIONARY OPERATORS

Jarosław Stańczak, Zbigniew Nahorski

Systems Research Institute, Polish Academy of Science,

01-447 Warsaw, ul. Newelska 6, Poland

E-mail: {stanczak, nahorski}@ibspan.waw.pl

Abstract
This paper describes some basic methods of specialized and heuristic genetic operators' design and shows advantages of approach that uses specialized methods of solution encoding and heuristic operators that are exactly adjusted to considered problem. As a computational example a problem of determining the optimal control signal to estimate parameters of dynamic object is used.
Keywords: evolutionary algorithms, heuristic operators.

1 Introduction

At the present stage of development of genetic algorithms (GA) it is obvious that applying traditional genetic operators (mutation or perturbation and crossover) to solve all possible optimization tasks is useless. Similarly binary encoding of all considered problems is not an unquestionable dogma. The tendency to use genetic algorithms in a manner as they were introduced by Holland [8] and Goldberg [6] to solve all problems lasted very long. But shortcomings of this approach have been seen very clearly since that and genetic algorithms have been changing from uniform method to solve everything to some kind of universal idea of evolutionary method specialized and adopted to specific problems. The evolutionary algorithm (EA) is more complicated, but its efficiency is much higher. To assess good results using EA it is necessary to design some essential elements of it:

- encoding of solutions;
- choosing genetic operators;
- choosing selection method.

Of course, it is possible to use some standard solutions, even the same as in GA, but specialized methods give significantly better solutions. Especially selection methods are usually more general in EA (there are problems that can be efficiently solved only using specialized selection methods, for instance optimization of non-stationary tasks) but do not particularly depend on the solved problem. That is why they are not considered in this paper.

Designing an efficient EA is a very interesting engineering challenge and a well-prepared EA will provide good and accurate results. Unfortunately, it is rather difficult to give some general algorithms, methods or criterions how to prepare efficient EA. However some standard approaches can be helpful in designing sufficient encoding and genetic operators for EA.

2 Encoding solution and designing genetic operators

The process of EA design must begin with accepting some method of encoding solution. It is obvious that the operators, which modify solutions, must take into account the data structure of the individuals. It should be emphasized that the accepted method of encoding solutions determines main properties of the designed EA, especially the accuracy of results and the rate of convergence. This comes from the fact that genetic operators depend directly on encoding solutions and their implementation and computational complexity is strictly connected with it. Thus proper encoding of solutions is very important and it is often necessary to test several methods before the best one will be found. Unfortunately it is usually difficult to foresee properties of EA with accepted encoding method. Theoretical aspects of this problem were considered in [1] and will not be discussed here. Only some rules connected with the problem of designing of genetic operators will be described. Arabas in his work [1] gives some strict requirements for encoding methods:

- each possible phenotype (solution of the problem) must have its equivalent in genotype space;
- the encoding method must not introduce additional extremes to the solved problem.

Sometimes it is possible to take into account additional requirements:

- automatically fulfill some constraints of the solved problem using proper encoding method;
- choose a method which doesn't require a lot of computer memory or big amount of computations to execute genetic operators or evaluate an encoded solution;
- applying the encoding method, which doesn't artificially enlarge the searching space of the problem.

It is usually difficult to foresee or to check all properties of designed encoding method and then it is necessary to verify the methods empirically.

In problems with continuous optimization we have basically two possible methods of encoding: binary or floating point, used as vectors or matrices, depending on the number of dimensions of solved tasks. Binary encoding usually gives smaller accuracy of computations (of course it is possible to use very long binary strings to assure similar accuracy as for floating point, but it works significantly slower than floating point encoding), thus the floating point encoding is used more frequently.

The case of discrete optimization is much more complicated, because usually there are many possibilities to encode the solutions of given problem:

- vectors of binary, integer, floating point values or more complicated data structures, sometimes of variable length;
- matrices of similar properties as vectors;
- dynamic lists of different kinds of objects;
- (binary) trees;
- some mixed structures of above mentioned elements.

Additionally individuals may contain some data required by specialized EA – some on-line tuned parameters of evolutionary computations, some flags or data used by operators, and data required by encoding method.

A possible encoding methods is to use diploid or polyploidy individuals ([7], [5]) to increase the flexibility and adaptation abilities of EA. This method is proper rather for simpler methods of encoding.

As it was mentioned earlier, binary encoding is not a universal method, best for all cases. It can be noticed that the binary encoding is not even similar to information encoding used by nature in DNA, where sequences of four, not two, elements are used. Thus the similarity to natural methods should not be a reason of too frequent application of binary encoding in EA.

Encodings mentioned in this paragraph probably do not exhaust the list of all possible methods used in EA. But they show the complexity of the problem of best individual encoding design to obtain fast and accurate evolutionary method of solving the given problem.

3 Methods of genetic operators design

To obtain a set of efficient operators we must realize that genetic operators and the way of encoding of the problem must be strictly combined with each other. Thus it is necessary to design these elements of EA together or taking into account accepted data structure for encoding solution.

Similarly to encoding methods there are some requirements for genetic operators ([1]):

- consistence of genotype space – it should be possible to generate every solution using the set of prepared genetic operators from any other solution in finite number of iterations;
- unbiasedness – operators should not prefer some directions in the search space[1].

[1] This requirement is not always necessary – probably all described in this paper heuristic operators do not fulfill this point, but work very well. It is likely that this statement should be replaced by: "the set of used genetic operators should be unbiased" or even weaker "the set of used genetic operators should contain also unbiased operators". This problem will be discussed later in this section.

The main purpose of genetic operators execution is to modify population members – solutions to search the solution space. Unfortunately, it is impossible to search the whole solution space in usual problems where EA are used. Thus, a set of genetic operators applied to solve the problem must be rich and diverse. Two simple operators, crossover and mutation, are often too weak to efficiently solve the majority of difficult tasks using EA. It is easy to show experimentally that EA with more than these two simple operators are almost always faster or more accurate. This fact can be easily explained. A set of several different operators modifies the population in the more diverse way, giving new, better and more scattered individuals. Additionally, heuristic operators have significantly higher probability of generating better offspring than simple, random operators, due to some embedded knowledge on the solved problem. Heuristic operators usually "know" which directions of searching are more promising and they prefer them, but this is what we call the "bias". But I am not sure, that the bias effect is always harmful. Of course it can be observed that some genetic operators, especially heuristic, loose (or exhaust) their searching abilities (they are biased i.e. limited to some directions of searching), when they are too intensively exploited, but a higher number of operators and/or a mixture of heuristic and random ones (mainly unbiased) manages this situation.

Let's conduct a simple experiment. Suppose that we want to solve a traditional TSP problem using EA. We have two genetic operators in our EA. One randomly changes the list of cities to be visited. The second (heuristic and biased) modifies the cities on the list in the manner that following cities are chosen from the nearest neighborhood of the several closest cities. If we use only the first operator in genetic computations, finding acceptable solutions would take a very long time. On the other hand if we use only the second one, we would obtain good solutions quite fast (probably better than using the first one in acceptable time), but after some very intensive progress in the beginning, the improvement of obtained results would almost stop. This is the effect of exhaustion of heuristic operator due to limitations of its search space. But if we use both of them, the results would be better than in the second case and the effect of exhaustion of heuristic operator won't appear, because of random modifications introduced by the first operator, which enable to escape from the trap of local search. Thus it is very important to have a set of different kinds of operators with heuristic – mainly biased and random – mainly unbiased ones.

Traditional genetic operators are divided into two groups with exploitation (crossover operation) or exploration (mutation/perturbation operation) properties. Specialized and heuristic operators are often hard to attach to one of these groups. The behavior of specialized operators is difficult to formalize and describe scientifically using mathematic formulae, but experiments show that they work very well.

It is also difficult to foresee what should be the probabilities of operator choice, because it is often not easy to realize what is a character of an actual operator (more exploratory or more exploitative). There are two ways to overcome this problem:

- experimental tuning of probabilities
- making them self-adaptive.

An exhaustive survey of methods used in parameter control in EA can be found in [3, 4, 9, 11, 14, 15, 16, 17] and also a short summary in the section 4.4 of this paper.

For the designing purposes genetic operators can be divided into two classes differing in the operation methods. The first class contains random operators, which do not have additional information about the solved problem. Of course, they must fulfill requirements of solved problem and encoding method, but they base on simple random changes of solutions. This group covers operators similar to the following:

- mutation – the action, which changes a value of a randomly selected position to another one from the set of possible values;
- crossover – the action, which exchanges parts of genotype between two or more individuals;

- inversion – the action, which places elements of solution in reverse order;
- concatenation, exchange, movement (or similar actions) of randomly chosen fragments of solutions within one individual;
- multiple versions of the mentioned operators

Random operators in most cases are very universal and can be applied in many problems without bigger changes. EA, which use exclusively random operators, are rather slow and sometimes cannot efficiently solve difficult problems [19].

The second group covers heuristic methods, which use an advanced knowledge about solved problem, or use simple optimization methods (hybrid optimization methods, like 2-opt, greedy or similar), sometimes with some randomly chosen parameters. Heuristic operators are almost always biased – they prefer some search directions and they "know" that this leads to better solutions. This is the main advantage in applying this kind of operators.

Several groups of heuristic operators can be selected:

- operators, which depend on current and/or former values of quality function of the solved problem;
- methods, which base on some common and simple or greedy methods of optimization;
- above methods specific for the solved problem.

The above mentioned methods probably do not exhaust all possibilities, but cover the most popular groups of heuristic genetic operators. Operators which use values of quality functions of the problem or can value committed modifications of solution genotype are rather universal and can be applied in many problems, except these ones, where a limitation on number of fitness function execution is imposed or valuation of solutions is computationally very expensive.

If it is possible to compare the values of fitness/quality function before and after modification of the solution, then it is possible to design an operator, which makes modifications several times. Each of them is valued and then the best one is accepted. If it is impossible to value modifications on line, sometimes it is possible to build operator which analyses old stored values of fitness function and committed movements and then prepare a new movement, better than a pure random one.

Another universal method of genetic operator design is to use elements of a simple optimization method. In the case of continuous optimization it can be several iterations of a gradient or similar optimization method. For discrete case we may use for instance "tabu-search", "simulated annealing", k-opt, some methods of local search or other "greedy" methods.

Methods designed for the actual solved problem are difficult to specify, because this class of operators may be very large and the only limitation for using them are very high computational complexity and lack of sufficient imagination of a designer.

Another issue is to design operators to make it possible to automatically consider limitations of the problem, or in different words to find operators that always generate feasible solutions. Probably this is not possible in every situation, but EA without applying expensive repair algorithms or punishment functions for constraints violation can work faster and more efficiently.

It is important to remember that it is always profitable to apply operators enriched with simple heuristic rules, which can increase the probability of achieving better solutions than in the case of pure random methods. But random methods are also necessary and the best results can be obtained combining both types of operators.

Next paragraph contains a simple example than can illustrate the idea of specialized EA with a designed set of heuristic and random operators.

4 EA designing example – determining the optimal „bang-bang" signal to estimate parameters of a dynamic object

In this example the aim is to find the optimal control signal, which enables to estimate the unknown parameter of a first-order dynamic object with the minimal error. The below sketch of the method is included to understand the role of EA in solving it. More details about this problem can be found in the papers [12] and [13].

4.1 Description of the problem

The considered first-degree dynamic object is described by the following differential equation:

$$T\frac{dx}{dt}+x(t)=u(t-t_d), \quad \text{with} \quad x(t_0)=0 \tag{1}$$

where:
T – unknown estimated parameter, $u(t)$ – input signal, t_d – pure delay, $x(t)$ – output signal.
A general solution of equation (1) has the following form:

$$x(t) = Ce^{-\frac{t}{T}} + \frac{1}{T}\int_{t_0}^{t_N} e^{-\frac{t-\tau}{T}} u(\tau - t_d)d\tau \tag{2}$$

In this problem we would use discrete, partially constant signals, which can change only in discrete moments, thus the solution (2) can be transformed to formula (3), considering zero initial condition (C=0):

$$x_n = \alpha x_{n-1} + bu_{n-k} \tag{3}$$

where:
$x_n=x(t_n)$, $\Delta=t_n-t_{n-1}$, $\alpha=e^{-\Delta/T}$, $k=t_d/\Delta+1$, $b=1-\alpha$.
The output signal is measured with some error ε_n:

$$y_n = x_n + \varepsilon_n \tag{4}$$

It is assumed that ε_n is a sequence of independent random variables with distribution $N(0,\sigma^2)$. Using the z transform it is possible to obtain the formula (5), which describes the model error:

$$e_n = y_n - \frac{1-\alpha}{1-\alpha z^{-1}}u_{n-k} \tag{5}$$

where:

the model error e_n corresponds to the measurement error ε_n.

To minimize the variance of the parameter estimation error the following Fisher information should be maximized or its inverse should be minimized [10], [20]. In this case the Fisher information matrix reduces to a scalar value, because the vector of estimated parameters contains only one parameter. After some transformations the following form of informational is obtained:

$$M(\alpha) = \frac{1}{\sigma^2} \sum_{n=k}^{N} \left(\frac{\partial e_n}{\partial \alpha} \right)^2 \tag{7}$$

where:

$$\frac{\partial e_n}{\partial \alpha} = \frac{1 - z^{-1}}{1 - 2\alpha z^{-1} + \alpha^2 z^{-2}} u_{n-k} \tag{8}$$

and N – number of measurements.

Finally, solving the differential equation (8) and putting it into (7), the following formula for the quality function of considered problem is arrived at:

$$\max F(\alpha) = \max M(\alpha) \bullet \sigma^2 = \max_{u_{n-k}} \sum_{n=k+1}^{N} \left\{ \alpha^n \bullet \sum_{l=k+1}^{n} \alpha^{-l} \left[1 + (n-l) \bullet (1 - \alpha^{-1}) \right] \bullet u_{l-k} \right\}^2 \tag{9}$$

where:
n – control step (n=1,2....N-k), u_n – control signal at the step n.

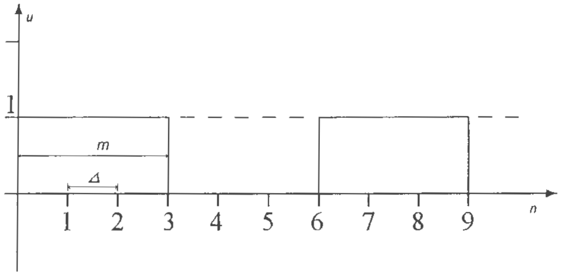


Fig. 1. The control signal.

It can be proved [12] that the optimal control signal is a square wave and without loss of generality of solution, minimal and maximal values can be accepted as "0" and "1" (Fig. 1). Optimal sequences of controls can be now found by minimizing function (9). It is impossible to find the optimal solution testing all possible combinations of "0" and "1" even for very short sequences. For example, for N=100 it would be necessary to compare 2^{100} values of function (9). Thus evolutionary algorithm seems to be a good method to solve this problem.

4.2 Evolutionary method with binary encoding

Sequences of controls contain only "0" and "1", thus it seems natural to accept a binary encoding method and such a method has been applied. Fig. 2 presents a member of population in the binary encoded individuals with N controls, always beginning from value "1".

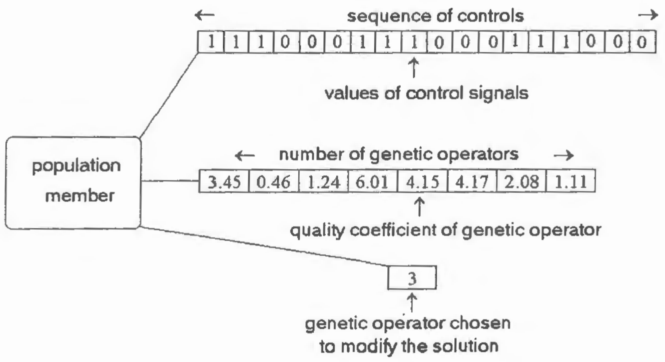

sequence of controls
1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0
values of control signals
population member
number of genetic operators
3.45 0.46 1.24 6.01 4.15 4.17 2.08 1.11
quality coefficient of genetic operator
3
genetic operator chosen to modify the solution

Fig. 2. Binary encoding of a population member.

Also a set of problem specialized genetic operators has been prepared:

- **mutation I** - typical binary mutation;
- **mutation II** – searches for places where controls change from "0" to "1" or "1" to "0" and performs negation of bits;
- **exchange** – random exchange of code fragments;
- **operation 2-opt** – uses a method of greedy optimization, called 2-opt ([18]) to improve the solution;
- **„intelligent" exchange** - random exchange of code fragments but performed only whet it gives an improvement of fitness function value;

- **operation „cycle detection"** – searches for the most frequent periods of cycles in the solution and modifies it to obtain a solution that consists only of these frequent values of periods.

Computer simulations showed that accepted encoding method with prepared set of genetic operators has several shortcomings. Computations lasted very long and results were rather inaccurate because long time of simulation caused stopping the algorithm at the maximal allowed computation time. This situation was caused by the fact that results obtained were nearly periodic with small perturbations and only changes of controls at ends of the adjacent periods introduced new important features. Modification of sequences of "0" or "1" in the middle of a period were mostly useless. Additionally, computing the values of fitness function with controls equal 0 is useless, because the result is always 0. Of course this computations can be easily eliminated, but the whole solution must be anyway checked bit by bit. Another disadvantage of presented approach is that binary encoding is also memory consuming for longer sequences of controls.

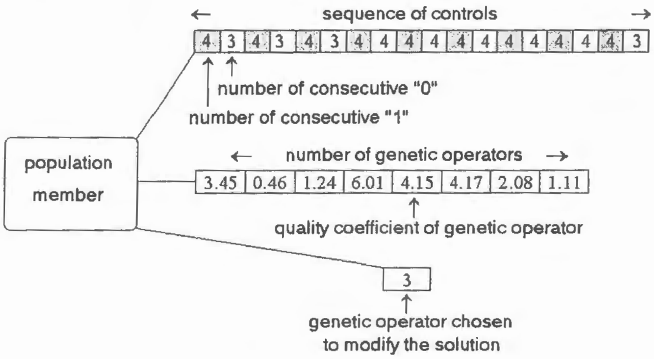
4.3 EA with integer encoding

Shortcomings of binary encoded EA directed our interest to other method. The conducted tests for integer encoding showed that this method gives similar results, but significantly faster. In this encoding the consecutive numbers of integers denote alternately periods of "0" and "1" sequences, always beginning with the sequence of "1" (it is useless to begin the identification experiment with signal "0", because it only causes additional delay). A scheme of population member with integer encoding is presented in Fig. 3. This encoding is a little more difficult than binary, because the length of encoding string is not

constant, and genetic operators must ensure that the number of controls equals *N*. But integer encoding brings some advantages. Encoding strings are usually shorter than in the binary case (only solution with all periods equal 1 would have the same length as binary version, longer periods would give shorter encoding strings), thus the algorithm uses less memory and works faster. Each modification in the solution changes only the period of some control sequence without inserting unnecessary values in the middle of the control sequence. It is also easy to skip computations of fitness function when a sequence of "0" is encountered.

Improved version of EA possesses also a different set of genetic operators:

- **exchange (single and multiple)** – random exchange of cycles;
- **movement (single and multiple)** – increases the length of one cycle at the cost of its neighbor;
- **insertion** of new cycle;
- **deletion** of randomly chosen cycle;
- **introduction** of new individual with randomly chosen lengths of cycles;
- **"intelligent" introduction** – several possible new individuals are prepared and the best one is accepted.



sequence of controls
4 3 4 3 4 3 4 4 4 4 4 4 4 4 4 4 4 3
number of consecutive "0"
number of consecutive "1"
population member
number of genetic operators
3.45 0.46 1.24 6.01 4.15 4.17 2.08 1.11
quality coefficient of genetic operator
3
genetic operator chosen to modify the solution

Fig. 3. Integer encoding of a population member.

The advantages of integer encoding are unquestionable, as compared to the results and shortcomings of the binary encoding. So it seems that traditional binary encoding or similar simple methods may be mainly useful at the first stage of designing efficient evolutionary method.

4.4 Management of genetic operators

Bigger number of specialized genetic operators requires applying some method of choosing among them in an iteration of the algorithm. In the approach used in [11, 15, 16, 17] it is assumed that an operator that generates good results should have bigger probability of selection 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. Thus, every individual has been given a vector of floating point numbers, beside encoded solution. Each number corresponds to one genetic operation. It is a measure of quality of the genetic operator (a quality factor). The higher the factor is, the higher is the probability of the operator. The ranking of qualities forms a base to compute the probabilities of appearance and execution of genetic operators. This set of probabilities can be treated as a "private" base of experience of genetic operators' efficiency of every individual and according to it, an operator is chosen in each epoch of the algorithm. Due to the gathered experience one can increase chances of its offspring to survive.

The method of quality factor computing is based on reinforcement learning [2] (an algorithm used in the machine learning). An individual is treated as an agent whose role is to select and call one of the evolutionary operators. Selection of the *i*-th operator can be regarded, as an agent's action a_i leading him to a new state s_i, which in this case is a new solution, modified by executed genetic operator. Agent receives reward or penalty respectively to the quality (fitness function) of the new state (solution). The aim of the agent is to perform the actions, which give the highest long term discounted cumulative reward V^*.

$$V^* = \max_{\Pi}\left(V^{\Pi}\right), \qquad V^{\Pi} = E_{\Pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\right\} \tag{10}$$

where:

Π - strategy of the agent, V^{Π}- discounted cumulative reward obtained using strategy Π, E_{Π} - expected value of reward using strategy Π, γ – represents discount factor, k – represents following time steps, t - represents current time.

The following formula can be derived from (10) and is used for the evaluation purposes. In the presented experiments the values of β and γ were set to 0.1 and 0.2, respectively.

$$V(s_{t+1}) = V(s_t) + \beta\left[r_{t+1} + \gamma V^*(s_{t+1}) - V(s_t)\right] \tag{11}$$

where:

$V(st)$ - is a quality factor or discounted cumulative reward, β - is a learning factor, r_{t+1} - represents the reward for the best action, which is equal to the improvement of the quality of a solution after execution of the evolutionary operator, t - current moment in time.

The quality coefficients can be easily converted into a vector of probabilities of evolutionary operators' execution, using simply normalization of its elements.

4.5 Results of computer simulations

Computer simulations proved that new version of EA with integer encoding works about 10 times faster than that using binary encoding, obtaining better solutions. This impressive increase of computation speed resulted from several factors:

- smaller encoding strings - less memory and operations needed to manipulate them;
- easy method to skip unnecessary computations;
- simpler and faster genetic operators;
- elimination of unnecessary degrees of freedom in encoding method.

The comparison of results of computer simulations ($\hat{m}$) and the approximate analytical estimations of control signal period (x) are presented in Tab. 1. The values of x and $\hat{m}$ show periods of consecutive "0" and "1" in the control signal. As it can be noticed in the Tab. 1, they show good similarity between approximate analytical estimations presented in [12] and results obtained using EA for higher values of ΔT, starting from ΔT=0.3. Bigger differences for longer periods are probably caused by numerical errors – for higher values of m extremely small numbers appear in computations, according to formula (9).

ΔT	α	X	$\hat{m}$
0,02	0,98	324,64	141
0,1	0,90	46,27	32
0,2	0,82	19,22	15
0,3	0,74	11,40	11
0,4	0,67	7,90	8
0,5	0,61	5,95	6
0,6	0,55	4,74	5
0,8	0,45	3,37	4
1,0	0,37	2,65	3
1,2	0,30	2,41	2
1,5	0,22	1,88	2
2,0	0,14	1,50	1

Tab. 1. Experimental results of the problem quality function optimization (α, ΔT – like in formula (9), x – result of an approximate analytical estimation of control signal period, $\hat{m}$ - the results of length of periodic control sequence obtained using EA optimization).

5 Conclusions

It can be clearly seen that the performance of EA depends mainly on accepted data structure in solutions encoding and also its consequences – specialized genetic operators. It is not always possible to design the best encoding method without experiments, because it is often difficult to foresee all properties of EA theoretically, without experiments and comparisons with other EA methods. Unfortunately, even simple specialized solution encoding with adjusted random and heuristic genetic operators are too difficult for theoretical analysis. The example presented in this paper shows that using binary encoding (or different simply method) to solve problems, even where it seems to be fully justified, may be misleading. But it may be treated as a universal "first step" to learn properties of the considered problem, which helps in finding better solutions. Sometimes several stages must be done, before good results are discovered.

References

[1] Arabas J. (2001). Wykłady z algorytmów ewolucyjnych, WNT, Warszawa, (in Polish).

[2] Cichosz P. (2000). Systemy uczące się, WNT, Warszawa, (in Polish).

[3] Davis L. (1989). Adapting Operator Probabilities in Genetic Algorithms, Proc. of the 3rd ICGA.

[4] Davis L. (1991). Handbook of Genetic Algorithms, Van Nostrand Reinhold.

[5] Goldberg D. E., Smith R. E. (1987). Non-stationary Function Optimization Using Genetic Algorithms with Dominance and Diploidy, II ICGA, Lawrence Erlbaum Associates.

[6] Goldberg D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley.

[7] Hadad B. S,. Eick C. F. (1997). Supporting Polyploidy in Genetic Algorithms Using Dominance Vectors, EP'97, vol. 1213 LNCS, Springer.

[8] Holland J. (1975). Adaptation in Natural and Artificial Systems, Ann Arbor, The Univ. of Michigan Press.

[9] Julstrom B. A. (1995). What Have You Done for Me Lately? Adapting Operator Probabilities in a Steady-State Genetic Algorithm, Proc. of the 6th ICGA, University of Pittsburgh, 1995.

[10] Kacprzyński B. (1974) Planowanie eksperymentów. Podstawy matematyczne, WNT, Warszawa (in Polish).

[11] Mulawka J. Stańczak J. (1999). Genetic Algorithms with Adaptive Probabilities of Operators Selection, Proceedings of ICCIMA'99, New Delhi, India.

[12] Nahorski Z. (1994). Planowanie eksperymentu dla identyfikacji modeli elementów hydraulicznych w oczyszczalni ścieków w Rzeszowie, Raport 19/9/S-10/96. IBS PAN, Warszawa (in Polish).

[13] Nahorski Z. Stańczak J. (2005). Optymalny sygnał typu bang-bang do estymacji parametrów w obiekcie pierwszego rzędu, Z. Bubnicki, R. Kulikowski, J. Kacprzyk, XV KKA, Tom I, IBS PAN, Warszawa (in Polish).

[14] Niehaus J. Banzhaf W. (2001). Adaptation of Operator Probabilities in Genetic Programming, Proc. of 4th EmoGP Conference, Como 2001, Springer, Berlin, 325-336.

[15] Stańczak J. (1999). Rozwój koncepcji i algorytmów dla samodoskonalących się systemów ewolucyjnych, PhD, PW (in Polish).

[16] Stańczak J. (2000). Algorytm ewolucyjny z populacją "inteligentnych" osobników, Materiały IV KAEiOG, Lądek Zdrój (in Polish).

[17] Stańczak J. (2003). Biologically inspired methods for control of evolutionary algorithms, *Control and Cybernetics*, 32(2), 411-433.

[18] Sysło M. M., Deo N., Kowalik J. S. (1983). Discrete Optimization Algorithms with Pascal Programs, Prentice-Hall Inc.

[19] Stańczak J. (2005). Optimal control of multistage deterministic, stochastic and fuzzy processes in the fuzzy environment via an evolutionary algorithm, *Control and Cybernetics*, vol. 34 No. 2, 525-552.

[20] Zarrop M.B. (1979). Optimal Experiment Design for Dynamic System Identification, Springer, Berlin.