Raport Badawczy
Research Report

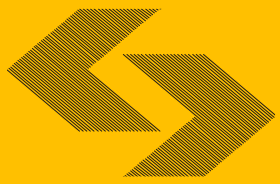
RB/89/2003

Biologically inspired methods for control of evolutionary algorithms

J. Stańczak

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences



Control and Cybernetics

vol. **32** (2003) No. 2

Biologically inspired methods for control of evolutionary algorithms

by

Jarosław Stańczak

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

Abstract: In this paper two methods for evolutionary algorithm control are proposed. The first one is a new method of tuning the probabilities of genetic operators. It is assumed in the presented approach that every member of the optimized population conducts his own ranking of genetic operators' qualities. This ranking enables computing the probabilities of execution of genetic operators. This set of probabilities is a basis of experience of every individual and according to this basis the individual chooses the operator in every iteration of the algorithm. Due to this experience one can maximize the chances of his offspring to survive.

The second part of the paper deals with a self-adapting method of selection of individuals to a subsequent generation. Methods of selection applied in the evolutionary algorithms are usually inspired by nature and prefer solutions where the main role is played by randomness, competition and struggle among individuals. In the case of evolutionary algorithms, where populations of individuals are usually small, this causes a premature convergence to local minima. In order to avoid this drawback I propose to apply an approach based rather on an agricultural technique. Two new methods of object selection are proposed: a histogram selection and a mixed selection. The methods described were tested using examples based on scheduling and TSP.

Keywords: genetic algorithms, adaptation, adaptive evolutionary algorithms.

1. Introduction

In the early years of evolutionary algorithms (EA) development much attention was devoted to showing their similarity to processes observed in nature (Holland, 1975). Such concepts as strong competition among individuals and struggle for survival were introduced. Also genetic operators, which modify the

genetic code of members of the population, were similar to natural ones. As it follows from experiments, traditional evolutionary mechanisms are not sufficient in many cases for a fast growth of the population in a desired direction. More complicated, adaptive methods are much better. There are big difficulties with the theoretical description of their behavior and properties, but there are first results, some of them very surprising: Gunter (1999) and Agapie (1999).

In the classic genetic algorithm, proposed by Holland (1975), both of the operators used, crossover and mutation, had constant probabilities of working, chosen intuitively or experimentally. Individuals of the population were coded as binary strings. The general rule was to assign a high probability $(0.8 - 1.0)$ to crossover and a low one to mutation $(0.01 - 0.2)$. These values were chosen not only intuitively, but were also based on biological and experimental data. Mutation is responsible for exploration of the search domain, while crossover exploits the previously found best regions. Too high level of mutation can lead to loss of convergence to the optimal solution, because of the fact that exchanges of the genetic material are stronger than process of evolution directed by fitness function. Because of this, the probability of mutation is very low. High level of crossover, however, intensifies exploration of local extrema and it is useful to keep its high probability.

Many problems cannot be effectively solved using traditional operators and methods of encoding. It is necessary to use specialized ways of encoding solutions as members of the population and genetic operators designed for operating on them. In that case it is difficult to foresee what values the probabilities of operator choice should have, because it is often not easy to realize what is a character of a given operator. There are two solutions for overcoming this problem: experimental tuning of probabilities or making them self-adaptive[1]. There were some trials of experimental evaluation of parameters of genetic optimization for several problems, using traditional genetic algorithm (DeJong, 1975), but it is rather impossible to investigate the whole domain of possible modifications of genetic algorithms for different problems. Of course, tuning of the parameters for a given problem is a very good (though time consuming) way of finding optimal values of them. The second possibility, however, is much more promising. One of the earliest solutions was to use a genetic subalgorithm to optimize the parameters of the main genetic optimization process – Grefenstette (1986). But it was a rather slow method. Next step was to find the probability of appearance of a genetic operator, connected with its behavior. Such methods were described in Cronen and Eiben (2001), Davis (1989), Davis (1991) and Julstrom (1995). They are based on the qualities of the operator, which modifies the population of solutions. Every operator gets (if chosen) its period of time, when it affects the population. So, all the modifications of an evaluation function can be assigned to a particular operator and they modify its probability of appearance.

[1] Exhaustive surveys of used methods of parameter control in EA can be found in Eiben, Hinterding and Michalewicz (1999), Smith and Fogarty (1997) or Schaefer (2002).

They are also backpropagated to the previously used operators, whose qualities are also taken into account and their probabilities are also changed. Thanks to this method any number of operators may be used, no matter how they work, randomly or on the basis of knowledge. The method described in Section 2 of this paper continues and develops the latter idea.

The second idea proposed in this paper deals with the problem of individual selection. The majority of used methods are based on a natural selection or similar, rather non-adaptive methods (a short description of some of them will be given later in this paper), which are good mainly for big populations with many individuals (ideally infinitely many). Examples taken from real life indicate that good results may rather be achieved by applying in the population agricultural measures as opposed to the principle of natural selection. It should be noticed that in nature each living organism is endangered by a number of harmful factors. Simultaneously it endeavors to adopt to a maximum extent to the environment. Thus, its objective function is uncommonly complicated. The influence of the environment and its disadvantageous effects are largely eliminated by man. Due to this fact, individuals with desired features are derived much faster than in nature. In such an approach the potential advantages of individuals can be utilized to great extent. However, such individuals could probably not survive in nature. This idea is the basis of the method of controlled selection presented in this paper.

2. A new method of probability control

2.1. Description of the method

In the approach presented it is assumed that an operator which generates good results should have higher probability and more frequently affect the population. But it is very likely that an operator, which is good for one individual, gives worse effects for another, for instance because of its location in the domain of possible solutions. Every individual may have its own preferences[2]. It is rather common situation in nature - every living creature has its own needs concerning environment of life, food, temperature, light, etc. The population of solutions created for solving technical problems has also biological origins and can probably develop better, if the preferences of its members are considered. The idea of personal preferences is realized as follows. Every individual has a vector of floating point numbers - q (besides the encoded solution). Each number corresponds to one genetic operation (the number of operators may vary during computations). It is a measure of quality of the genetic operator. The higher the number, the higher the probability of execution of the operator.

[2]The idea of "personal preferences" of each population member is also used in Krink and Ursen (2000) but with a quite different method of adaptation.

This relationship may be written as follows:

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

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).

The algorithm of this method is provided below:

1. Initialization of the population of solutions and the starting values of q_0;
2. i:=1;
3. Selection of genetic operators (and partners if necessary) by the members of population;
4. Modification of individuals using selected genetic operators;
5. Evaluation of new values of fitness function and new values of q_{ij} for all members of population;
6. Evaluation of new probabilities of operators selection[3];
7. Selection of members of the offspring population;
8. i:=i+1;
9. If stop condition not satisfied, then go to 3;
10. End.

This method can be applied both in the case of operators changing one individual (like mutation) and of those affecting two or more individuals (like crossover). In the first case, there is no problem in executing operators, as this depends only on personal preferences. In the second case, two (or more) individuals must choose the same operator to make it executable. It is necessary to implement a mechanism of searching for population members with the same preferences and this phenomenon is very similar to situations from nature.

The parameters q_{ij} are closely related with values of the fitness function or, more precisely, with changes of this function resulting from the activity of the genetic operators. It is possible to attribute every change of fitness function to a given operator, because only one operator modifies one member of population in one generation. Of course, during one generation different individuals use different operators, but one individual is changed only by the chosen operator. The formula (2) shows how the quality of the operator is obtained.

$$q_{ij}(t+1) = \begin{cases} q_0 + \frac{x_{ij}(t)}{f(Q(t),\bar{x}_j(t))} + \alpha * q_{ij}(t) & \text{for i=1} \\ q_{ij}(t) & \text{for other i} \end{cases} \tag{2}$$

where:

[3]In real use it is not necessary to compute these probabilities, because the process of operators' selection may use vectors of qualities without computation of probabilities.

$q_{ij}(t)$, $q_{ij}(t+1)$ – quality of the i-th operation for the j-th individual in consecutive generations;
l – number of the chosen operator;
q_0 – a small credit value[4];
$f(..)$ – normalizing function, its arguments can be:
$Q(t)$ – the best solution at the moment t (the quality function value);
$\bar{x}_j(t)$ – the mean value of improvements of the quality function of the individual j (and its ancestors) during evolutionary computations;
$x_{ij}(t)$ – an improvement of the problem's quality function, obtained by the i-th operation for the j-th member of the population, defined as follows:

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

$Q_{ij}(t)$ – solution of the j-th individual, obtained using i-th operator;
α – a coefficient of forgetting $\in (0, 1)$.

The first element of the formula (2) – q_0 plays a role of a 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 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 improvement of the fitness function.

The second term in addition is a normalized value of an improvement of the problem's quality function in the current generation. The improvement of the quality function can be taken into account in two ways: improvement of the globally best solution (which is more desirable) and improvement of the offspring in relation to its parents. Both possibilities can be used in formula (2) with appropriate weights (higher for the first and lower for the second way). The normalization function is responsible for making changes of the quality function independently of the character of the problem. It is easy to imagine a problem where a small change of quality function (for solutions found in consecutive generations) is for instance 10^5 and another one, where this value is typically 10^{10} times smaller. This situation would require a long process of tuning the parameters of the formula (2), if the normalization function were not applied. After adding normalization the range of searched optimal values of parameter q_0 and α can be significantly decreased. The exact form of the normalization function is not given in formula (2). It is possible to use any function that transforms the values of quality function improvements into the numbers from the range $(0, 1)$ and higher positive changes are kept higher after

[4]Methods of obtaining values of parameters for the formula (2) are discussed in subsection 2.2.

normalization. The successfully tested functions are: the best value of quality function reached by optimization process and the mean value of improvements during the algorithm simulation. More information about normalizing function will be given further in this paper.

The third part of the formula (2) is responsible for storing previous achievements of an operator multiplied by a forgetting factor α. The parameter α is responsible for balancing the influence on the quality factor of an operator from old and new improvements. It should be noticed that some genetic operators may achieve good results in some phase of simulation, and then exhaust their capabilities. On the other hand, the remaining operators, probably better in the subsequent phases, would have small probabilities of appearance, if the factor α had not been introduced. So, it would take a lot of time to change this situation and the process of genetic computations would be slowed down. The effect of forgetting former achievements can overcome this problem. When operators do not change the globally best solution for some time, the probabilities of operators become small. After every generation only the value of q_{ij} bounded with the chosen operator is updated, the other ones remaining unchanged. Only one operator is executed in one generation for one individual, so there is no reason to change the coordinates corresponding to the other, not selected operators. Setting the value of α to 1 can cause the situation of domination of operators, which do not produce any profit, but were good in the early stages of evolutionary process and consequently slow down computations. On the other hand, the value of 0 causes the situation where the quality of operator relies only on the most recent information. It gives the situation of almost equal probabilities of appearance of all operators, because significant improvements of quality function are rare and all operators' qualities would have the same values during the major part of computation. In the case of prolonged lack of positive achievements of an operator, its value q_{ij} is established at the level described by the formula (3):

$$q_{ij}(\infty) = \lim_{t \to \infty} q_{ij}(t) = \frac{q_0}{1 - \alpha} \tag{3}$$

where:
$q_{ij}(\infty)$ – a limit value of $q_{ij}(t)$;
all other symbols being like in the formula (2).

It is obvious that formula (2) reduces itself to a sum of infinite convergent geometric sequence ($\alpha < 1$) when the profit is zero for a long time.

2.2. Parameters for practical use

The formula (2) requires two numerical parameters q_0, α, and a normalizing function. The process of selecting these elements is not very complicated and is described below. Generally, the parameter q_0 ought to be smaller than the important changes of normalized quality function. The parameter α should

belong to the range $(0, 1)$ to assure convergence of geometric sequence, generated by formula (2). The span of the practically used values of α is narrower $(0.6, 1)$. Too small values of α cause lack of positive feedback between the high quality of an operator and its frequency of appearance, because all qualities quickly become equal for all operators. Too big values cause practically elimination of worse operators at the beginning of the evolution, though they might become very useful later. Even selecting $\alpha = 1$ should not bring overflow errors (not convergent geometric sequence), because computations last a finite number of iterations.

It is also important to assure non-zero starting values of q_{ij} $(q_{ij}(0))$, because probabilities corresponding to operators cannot be zero. In the case of qualities equal zero only one operator chosen randomly or manually at the beginning of simulation would appear all the time until the end of evolutionary computations. A good idea for setting the initial values of q_{ij} is to use the formula (3). This value is achieved when no improvement is detected and that situation appears at the beginning and also at the end of simulation.

Beside the factors α and q_0 the third unknown parameter in the formula (2) is the normalizing function. Its role is to make changes of problem's quality function $(x_{ij}(t))$ independent of the specific problem. It makes possible to choose values of α and q_0 more universally. The simplest way of solving this problem is to take the best-found solution as a normalizing function. But this has some shortcomings:

- in the case of negative values of quality function it is necessary to use a modulus of the function:
$$f(Q(t), \bar{x}_j(t)) = |Q(t)| \tag{4}$$
where:
$Q(t)$ – the best solution at the moment t;
$\bar{x}_j(t)$ – the mean value of improvements of quality function;
- in the case the quality function approaches zero, its normalized value could infinitely increase. Scaling and translation of the quality function can overcome this problem (parameters a and b used in (5) are strictly connected with the specific problem):
$$f(Q(t), \bar{x}_j(t)) = b + aQ(t) \tag{5}$$
where:
$Q(t)$ – the best solution at the moment t;
$\bar{x}_j(t)$ – the mean value of improvements of quality function;
a, b – chosen coefficients;
- in the situation of a change in the sign of the quality function, the method described above can fail. In that case a good solution is to use the mean value of improvements of the quality function, calculated separately for each individual of the population:
$$f(Q(t), \bar{x}_j(t)) = \bar{x}_j(t) = \frac{\bar{x}_j(t-1) * (t-1) + x_{ij}(t)}{t} \tag{6}$$
where:

$Q(t)$ – the best solution at the moment t;
$\bar{x}_j(t)$ – the mean value of improvements of quality function;.
The latter case seems to be the most interesting. The main advantage of this function is adaptation. The mean value of improvements decreases during computation and also possible improvements become smaller on more advanced stages of evolution. This effect compensates for the decreasing intensity of evolution during computations.

The formulae (3) and (2) may be also used as a basis for determination of values of α and q_0. It is possible to accept the assumption that the quality factor of an operator can change its value in the first iteration by not more than 1.0 (using normalizing function as in formula (6)[5]):

$$\frac{x_{ij}(t)}{f(Q(1), \bar{x}_j(1))} \leq 1.0. \tag{7}$$

This should be an important change in relation to the coefficient $q_{ij}(\infty)$ amounting to, for instance, its 50%, which leads to the following dependence:

$$\frac{q_0}{1-\alpha} * 0.5 = 1.0 \tag{8}$$

where all symbols in (7) and (8) have the same meaning as in (2) and (3).

Assuming the value of $\alpha = 0.9$[6] we obtain $q_0 = 0.2$ and $q_{ij}(\infty) = q_{ij}(0) = 2.0$. The here described way of finding the parameters of the formula (2) is only a rough scheme, but it gives satisfying values. Generally, the method of adaptive selection of operators is resistant to the selected values of parameters. They can be selected from a wide range without any detrimental effects because the main role in valuing the operators is played by their behavior.

It is also possible to establish vectors of values α, q_0 and $q_{ij}(\infty)$, with coordinates specific for every operator. When there is an "a priori" information about the behavior of operators there may be a situation where different values of parameters for the operators may be used, but in most cases it is not necessary because the main role is played by the achievements of operators.

3. The non-controlled and controlled selection methods

3.1. A brief survey of traditional selection methods

In traditional evolutionary algorithms the following methods of non-controlled selection are used (Michalewicz, 1996, Goldberg, 1989):

- The roulette method. Individuals of the offspring population are selected in accordance with probabilities which are proportional to the values of

[5]In that case the mean value of improvements equals the value of improvement.
[6]This value is chosen, based on conducted experiments with various values of the parameter α.

their fitness function. An expected value of the offspring for the member of the parent population can be written as:

$$E_l(t+1) = \mu * \frac{F_l(t)}{\sum_{j=1}^{j=\mu+\lambda} F_j t} \tag{9}$$

where:
$E_l(t+1)$ – an expected value of offspring of l-th individual in the $t+1$ iteration;
μ – the size of parent population;
λ – the size of offspring population;
$F_l(t)$, $F_j(t)$ – value of fitness function for the l-th or j-th individual in the iteration t.

- The method of the best individual selection. In this method the following formula is applied:

$$n_l(t+1) = \begin{cases} 0 & \text{for} \quad F_l(t) - F_{min}(t) < 0 \\ 1 & \text{for} \quad F_l(t) - F_{min}(t) \geq 0 \end{cases} \tag{10}$$

where:
$F_{min}(t)$ – all individuals, whose fitness exceeds this value, enter the next population;
n_l – the number of offspring for the member l;
all other symbols are the same as in formula (9).

- The method of deterministic roulette is described by the function:

$$n_l(t+1) = f\left(\mu * \frac{F_l(t)}{\sum_{j=1}^{j=\mu+\lambda} F_j(t)}\right) \tag{11}$$

where:
$n_l(t+1)$ – number of offspring of l-th individual in the next generation;
$f(..)$ – a function converting a real value to an integer value;
μ, λ, $F_l(t)$, $F_j(t)$ – have the same meaning like in formula (9).

- Selection by stochastic remainder method with repetitions is a kind of combined method. Its first phase is based on the deterministic roulette method (11). The free places remaining after the first part are filled using the fractional parts of "individuals" by traditional roulette method, as it is described by the formula (12).

$$E(n_l(t+1)) = f(x_l(t)) + \left(\mu - \sum_{j=1}^{j=\mu+\lambda} f(x_j(t))\right) * \frac{x_l(t) - f(x_l(t))}{\sum_{j=1}^{j=\mu+\lambda}(x_j(t) - f(x_j(t)))} \tag{12}$$

where:
$x_l(t)$, $x_j(t)$ – an expected value of offspring in the roulette selection (9), denoted x_l or x_j for shortness;
$f(...)$ – a function, which returns an integer part of the argument;
all other symbols being like in formulae (9) and (11).

- Tournament selection is the method, in which μ times a competition between two (sometimes more) randomly chosen individuals is conducted

and a better one is associated with the descendant population. The expected value of the l-th individual offspring is shown in formula (13):

$$E_l(t+1) = \mu * \frac{(\mu+\lambda-l+1)^k - (\mu+\lambda-l)^k}{(\mu+\lambda)^k} \tag{13}$$

where:

$E_l(t+1)$ – an expected value of offspring of l-th individual in the $t+1$ iteration, $l \in (1..\mu+\lambda)$; μ – the size of parent population; λ – the size of offspring population; k – the size of the tournament (number of randomly chosen individuals, from which the best is a winner).

This list does not exhaust the whole domain of different selection methods; there are other very interesting ideas, like the ranking selection (Baker, 1985 and Goldberg, Deb and Korb, 1991) and the genetic algorithm with varying population size (Arabas, Michalewicz and Mulawka, 1994), and many others. The traditional roulette approach in stationary evolutionary algorithm gives usually poor results. After a long time of computations it stabilizes oscillating about some value far from the global extremum. Poor properties of the roulette method were criticized in very early works on genetic algorithms (DeJong, 1975). Much better results may be achieved using the selection of best individuals. However, this method does not produce a high level of pressure for the population development. The deterministic roulette method introduces a very strong pressure for the population development but it quickly loses the diversity of population and consequently the algorithm terminates at a far local extremum. The selection by stochastic remainder with repetitions is very similar to deterministic roulette but behaves better, because it assures a higher level of population diversity.

Having analyzed the traditional methods of selection we can conclude that they can not be controlled nor adapted. Thus, they do not make it possible to influence the process of selection and development of population by some external factors. The histogram and mixed selection methods presented in this paper allow for the fulfillment of this condition.

3.2. The new approach

3.2.1. Histogram selection

In the histogram selection, described by the formula (14), a list of individuals of different values of the fitness function is created (this list resembles a histogram):

$$n_l(t+1) = f\left(\mu * \frac{F_l(t)}{\sum_{j=1}^{j=s} F_j(t)}\right) \tag{14}$$

where:

s – number of values on the list;

all other symbols have the same meaning as in formula (11).

This list is usually shorter than a number of individuals in the population, because there are usually repetitions of the same solution and this fact is considered only once in the list produced by the histogram selection. Next, the mean value of the fitness function is calculated using only once each value from the list, no matter how many individuals are connected with this value. Each individual (or rather value from the list) propagates to the offspring population the corresponding number of individuals. First, the particular fitness function is divided by a sum of all values from the list. Next, this quotient is multiplied by a size of base (parent) population and finally rounded to the nearest integer value. In the case the calculated number is lower than the size of base population, an appropriate number of best creatures that were rejected in the first phase is added to the population. In the opposite case some of the worst individuals are removed. Fluctuations of the obtained size of the base population are caused by approximation of calculated real values by integers.

The method of histogram selection delivers a number of interesting features. Selection is carried out in a deterministic way, so that there is no possibility of the best individuals dying out, which can occasionally happen in the probability based methods. It also enables the maintenance the population diversity in a simple manner. In this approach the mean value of the fitness function depends only on the values of this function existing in the population, not on the number of appearances of these values in the population. Thus, the best individuals are not advanced excessively. Worse individuals have also got a chance to be selected to the new population. However, there is a possibility of losing good individuals described by the same fitness function values but with completely different genetic code. Thus, histogram selection may be extended so as to distinguish solutions having the same values of fitness function but representing different individuals. In the case of discrete function optimization, the simplest manner to distinguish individuals is to check if their genetic codes differ at least on a single position. When optimizing continuous functions it is also possible to check the similarity of individuals. In this case one should introduce some neighborhood of the respective point in which two solutions are considered identical.

3.2.2. Mixed selection

As follows from the previous point, histogram selection operates effectively by not allowing for a too early convergence of the algorithm. Its characteristic feature is the lower level of selective pressure towards promoting the best individuals than in the deterministic roulette. On the contrary, the deterministic roulette method outstandingly selects the best solutions, which results in rapid loss of population diversity and consequently – premature convergence. Thus, these two methods have different faults and advantages and a method connecting good features of these two methods can work better than each of them separately. The mixed selection is a solution, which has advantages of

both methods. This method consists of two parts: histogram selection and deterministic roulette, which are selected in random during the execution of the algorithm. The performance of this method is explained in Table 1.

Table. 1. An example of mixed selection performance (μ-parent population, λ-off spring population).

Iteration	μ	λ	μ	λ
	Histogram Selection		Deterministic Roulette	
1	5, 5, 4, 3, 2	1, 2, 3, 1, 4	5, 5, 4, 3, 2	1, 2, 3, 1, 4
2	5, 5, 4, 3, 2	0, 2, 3, 2, 4	5, 5, 4, 4, 3	0, 2, 3, 2, 4
3	5, 5, 4, 3, 2	1, 2, 3, 2, 4	5, 5, 4, 4, 4	1, 2, 3, 2, 4
	Deterministic Roulette		Histogram Selection	
4	5, 5, 4, 4, 3		5, 5, 4, 3, 2	

In Table 1, an example of certain fitness function maximization is considered. Two cases of possible sequences of appearance of the component selection methods are provided. Particular fields of the table comprise the values of the fitness function, where each number corresponds to one individual. To simplify and better illustrate advantages as well as faults of two selection methods, lack of improvement in this small part of the algorithm performance was assumed. Such situation occurs very often during computations. In the example of Table 1 the population size was $\mu = \lambda = 5$ for the strategy $(\mu + \lambda)$. The above hypothetical example of evolutionary computations shows the characteristic properties of the proposed method. The deterministic roulette selection exhibits the tendency to fill up the parent population with almost identical and best individuals[7], while the histogram selection assures to preserve the parent population diversity. As follows from this example, histogram selection also has the property of repairing the composition of too uniform parent populations. The two versions of selection (histogram and the deterministic roulette) supplement each other.

It is possible to control the properties of the mixed selection operation by introducing the probability of appearance of a particular selection version. The more frequent appearance of the roulette selection causes higher pressure towards promotion of the best solutions, which can in some cases (when there is a high level of population diversity) speed up the computations. On the contrary, histogram selection increases the level of population diversity, paying for it by the weaker pressure towards the promotion of the best individuals. Due to this mechanism it is possible to examine a given search area more effectively.

This approach allows the adjustment of features of the selection operation to the current requirements of the population (to preserve the diversity of the population or to intensify the selection pressure). The adaptation of probabilities

[7]It can be seen for instance in the third iteration of this method for the parent population, which consists of only two kinds of individuals.

of selection operations may be derived from statistical properties of the population. An idea of this method is shown in formula (15) (in all the equations below there is $p_h = 1 - p_d$ – because one of selection methods must appear).

$$p_h(t+1) = \begin{cases} p_h(t) * (1-a) & \text{for } max(F_{av} - F_{min}, F_{max} - F_{av}) > 3\sigma(F) \\ p_h(t) * (1-a) + a & \text{for } max(F_{av} - F_{min}, F_{max} - F_{av}) < 0.5\sigma(F) \\ p_h(t) * (1-a) + 0.5a & \text{for } \begin{cases} 0.5\sigma(F) \leq max(F_{av} - F_{min}, F_{max} - F_{av}) \\ max(F_{av} - F_{min}, F_{max} - F_{av}) \leq 3\sigma(F) \end{cases} \end{cases} \tag{15}$$

where:

p_h – probability of histogram selection appearance ($1 - p_h$ is the probability of deterministic roulette method p_d);

$max(...)$ – a function which returns a bigger value of its arguments;

F_{av}, 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_h.

If particular individuals are described by a too small standard deviation of the fitness function ($\sigma(F)$) with respect to the extent of this function ($max(F_{av} - F_{min}, F_{max} - F_{av})$), then it is desirable to increase the probability of appearance of histogram selection. On the contrary, when the diversity of the population is sufficient, the probability of the deterministic roulette selection can be increased to possibly speed up the evolutionary computations. If 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. In the experiments the statistic parameters of the population: 0.5 and 3.0 have been found empirically because they yielded the best results. The value of the parameter a has been fixed to 0.05. The method of tuning the parameters of the selection operation has also been practically tested. It has been found that this approach is better than selection with constant (even if selected best) values of p_h and p_d.

4. Simulation results

4.1. Solved problems

To assess the usefulness of the proposed methods a number of experiments were performed. Tests were carried out for widely known traveling salesman problem (TSP) of 1002 cities and for scheduling of time-dependent jobs on a multiprocessor system (1000 tasks).

Solutions for TSP were encoded as lists of cities to be visited in an order described by the list. To solve this problem several genetic operators were

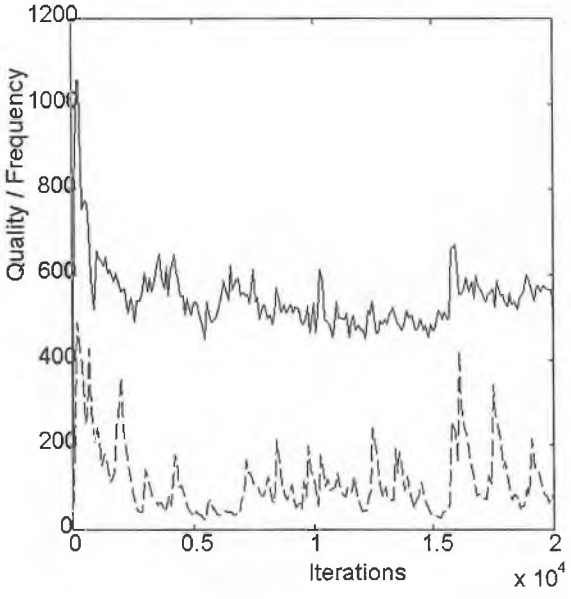
used: "problem blind"- random and heuristic. All operators were designed to assess problem limitations – the offspring always encoded valid solutions. The operators were:

- mutation – a random exchange of two cities in the list of cities;
- crossover – starting from the first city of one list, next cities are chosen in turn from one or second list (closer to the previously accepted city is selected from the parent individual and introduced to the offspring), preserving limitations of the problem;
- inversion – a fragment of the list is used in the reverse order;
- transposition – a fragment of the list of cities is moved to another place in the list;
- 2-optimal method – exchange of two chosen edges in the route, if it gives a shorter route (based on the widely used r-optimal method (Syslo, Deo and Kowalik, 1995) for approximate solving of TSP);
- neighborhood operator (two versions) – exchanges a city in the route for other, chosen from the list of the closest ones in the geometric sense (a list of several closest cities is generated for every city during initialization of the algorithm). The "neighborhood-1" operator simply exchanges a city from the list (found on the path) with a randomly chosen one. The "neighborhood-2" operator takes a city close to selected one from the path, moves all cities between them by one position an inserts the picked city next to the selected one.

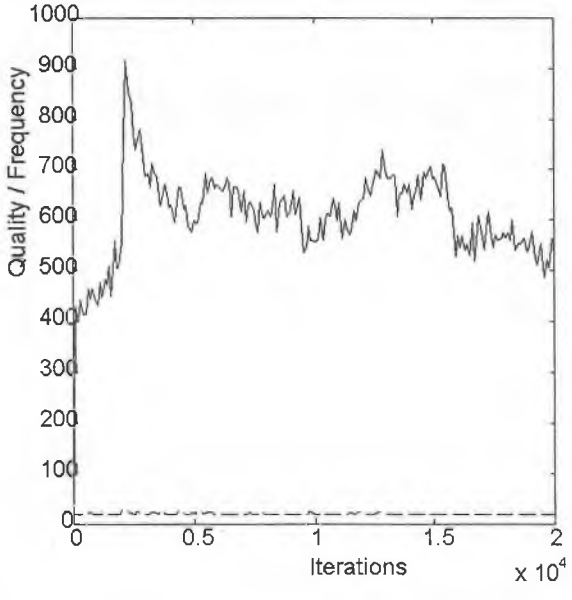
The members of the population for scheduling were coded as lists of tasks waiting for processing. A graph of time-dependencies was also introduced into the algorithm in the form of labels of waiting tasks. All operators are designed to assess valid solutions. Only "problem blind" operators were used (similar to those used for TSP): mutation, inversion and transposition.

4.2. The probability control

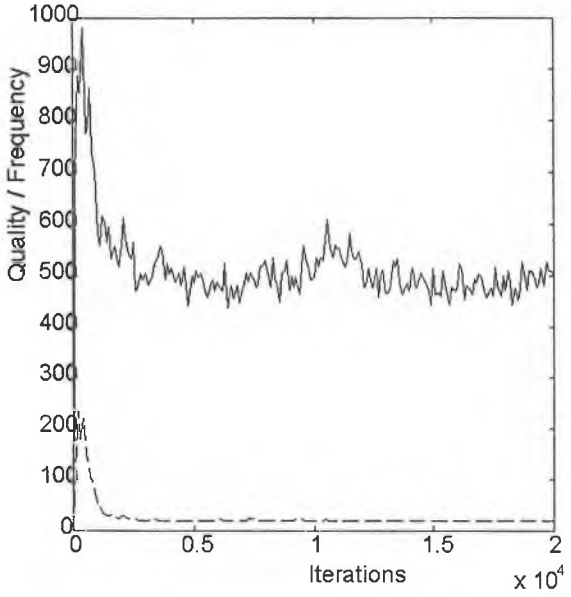
Fig. 1 shows dependence between the quality of an operator and its frequency of appearance for selected operators (only TSP). All data are collected from the whole population (not from one member) and that is why there is no exact and linear proportion between quality and frequency. Fig. 1b shows a very big jump of frequency and very small increase of quality – it can be caused by an important improvement of fitness function of one member, which has many "children" and these children also have chosen this operator. It would be difficult to show data from one individual during the iterations, because it lives only for one epoch. On the other hand it wold be difficult to trace the offspring of one, chosen individual, because it may die out or to the contrary, there can be too many of its children.



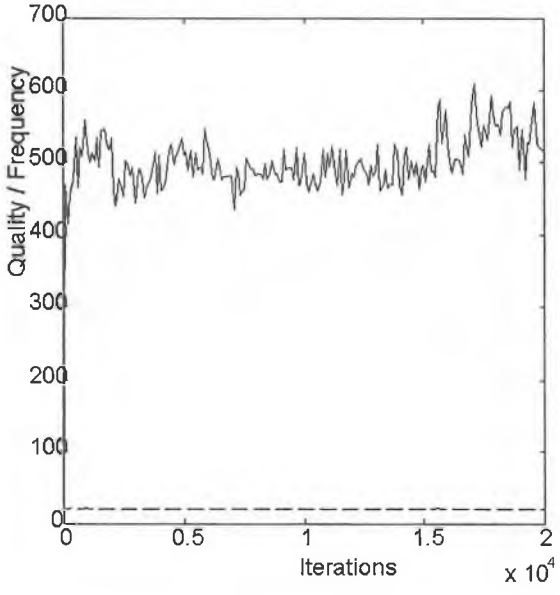
a) crossover



b) "neighborhood-2"



c) 2-optimal



d) "neighborhood-1"

Figure 1. The dependence between a quality factor of an operator (at the bottom of every picture) and its frequency of appearance (TSP – 1002 cities).

By analyzing these pictures it clearly can be seen that a better behavior of an operator results in an increase of frequency of its appearance and consequently speeds up evolutionary computations (which can be seen from Tables 2 and 3). Operators, which do not bring any contribution are not eliminated, but have approximately constant frequency. It is very likely that they behave in a way like mutation does and have exploration properties. They are very important, but they cannot appear too often, due to the possible effect of losing convergence to the optimal solution.

In Table 2 the influence of the factor α on genetic computations can be observed. Data from 75 simulations are collected and their mean values are presented for different numbers of iterations and values of factor α.

Table 2. A comparison of data obtained from 75 simulations (mean value) for several numbers of iterations and values of forgetting factor a (TSP). The row labeled "const" shows results obtained using the same probabilities for all operators.

	0	100	500	1000	2000	5000	10000	20000	50000
const	293042	289893	282191	278390	275278	272748	271492	270593	270376
α=0.0	293232	289193	280809	277246	274744	272686	271683	270999	270258
α=0.1	293050	288291	280065	276903	274605	272558	271527	270937	270246
α=0.2	293085	288247	279505	276229	274006	272152	271251	270686	269956
α=0.3	293096	288066	279729	276762	274455	272553	271648	271148	270453
α=0.4	293036	287667	279345	276256	273981	272188	271373	270705	269973
α=0.5	293072	287452	278798	275965	273765	271934	271257	270885	270236
α=0.6	293312	287840	278448	275956	274019	272234	271446	270972	270222
α=0.7	292973	286751	278052	275718	273837	272175	271579	271221	270671
α=0.8	293120	286729	277116	275137	273309	271715	271187	270891	270161
α=0.9	292916	285513	276244	274346	272815	271430	270984	270682	270154
α=1.0	292968	284877	275277	274980	274299	272262	271030	270588	270121

All simulations were started from solutions obtained from a simple algorithm that generates a route taking the city closest to the last visited one. When starting from randomly chosen cities it gives different solutions and works much faster than random initialization of the solution population. The strategy $(\mu+\lambda)$ was used with $\mu = \lambda = 60$[8]. The optimal solution for the considered task is known and equal 259045. The best-found solution by described program is by 3% worse after 10^6 epochs.

Considering the data from Tables 2 and 3 it is easy to see that the best results and the fastest computations can be obtained using $\alpha \in (0.6, 0.9)$. It should be noticed that setting $\alpha = 0$ does not result in equal values of probabilities for all operators because it does not eliminate the influence of their behavior. It only provides for the shortest period of remembering of their achievements, which lasts till the next execution of that operator.

[8]Arabas (2001) suggests that the best results of evolutionary computations can be obtained using $\mu/\lambda = 7$, but also adds that it can strongly depend on the given case.

Table 3. A comparison of data obtained from 25 simulations (mean value) for the selected numbers of iterations and values of the forgetting factor α (scheduling of 1000 tasks). The row labeled "const" shows results obtained using the same probabilities for all operators.

	0	50	100	200	500	1000	2000	5000	10000	20000
const	2342.1	2125.7	2036.0	1924.7	1740.3	1588.4	1445.5	1301.4	1232.0	1190.6
α=0.0	2348.6	2128.4	2036.6	1921.6	1744.8	1591.7	1450.3	1304.6	1234.2	1190.7
α= 0.1	2357.6	2134.5	2042.8	1927.6	1739.8	1590.2	1444.5	1301.8	1232.2	1191.1
α= 0.2	2345.1	2136.5	2042.5	1927.6	1742.8	1594.5	1450.9	1305.1	1235.2	1192.1
α= 0.3	2353.9	2121.1	2034.6	1917.3	1732.4	1588.0	1446.9	1302.3	1232.0	1190.9
α= 0.4	2339.0	2119.3	2024.1	1923.8	1744.8	1593.1	1450.4	1304.8	1235.3	1191.5
α= 0.5	2342.5	2138.9	2046.9	1921.2	1744.4	1593.9	1453.5	1301.4	1232.2	1191.1
α= 0.6	2332.3	2108.6	2014.3	1906.9	1732.0	1582.9	1447.9	1299.9	1231.3	1189.7
α= 0.7	2337.4	2115.2	2024.9	1911.4	1736.7	1590.0	1451.1	1303.3	1235.1	1190.8
α= 0.8	2331.5	2124.9	2025.9	1915.3	1730.9	1588.8	1450.0	1305.4	1233.7	1190.6
α= 0.9	2326.8	2097.4	2006.0	1894.9	1715.9	1571.0	1441.7	1300.1	1234.3	1190.9
α= 1.0	2344.5	2128.4	2022.7	1888.9	1686.0	1532.1	1413.5	1329.2	1290.3	1257.6

It should be noticed that the biggest influence of the adaptation method takes place at the early stages of the evolutionary computations (0-10000 iterations for TSP and 0-2000 iterations for scheduling). This is because the process of evolution is fast then and the operators bring a lot of information about it. This information enables to choose operators effectively. At the late stages of the evolution process the improvements of individuals are rare and the qualities of operators become almost equal. That is why the results are similar to the ones of the method without probability tuning.

The parameters of simulation for scheduling are the same as in TSP. Optimal solution is unknown, the best found one equals 1121 (after 10^6 epochs). Optimal values of α are from the same range as before, but differences between solutions for various α are bigger than in the case of TSP. This fact is related to the operators used. For TSP both "blind" and knowledge-based operators were used and they played the main role in computations. In scheduling only randomly working operators generated improvements and properly tuned probabilities of their appearance are much more important than in TSP.

4.3. The method of controlled selection

The empirical analysis of selection operation has been carried out for the two described problems. An analytical description of effects of the selection operation on the quality of the evolutionary algorithm is very difficult. Therefore, it should be done experimentally. The results of simulations are provided in Table 4.

Table 4. Results obtained for different methods of selection: I – classical roulette, II – deterministic roulette, III – selection by stochastic remainder method with repetitions, IV – tournament selection, V- best individuals selection, VI – histogram selection, VII – mixed selection with constant probability ($p_h = 0.45$ and $p_d = 0.55$), VIII – mixed selection with automatic tuning of probabilities.

Scheduling of time dependent jobs								
Iteration	I	II	III	IV	V	VI	VII	VIII
0	2341.1	2355.4	2350.1	2331.9	2336.3	2353.7	2357.0	2357.3
100	2154.7	2064.6	2058.7	2108.0	2062.8	2091.5	2075.7	2082.1
500	1961.4	1760.0	1767.7	1821.2	1767.8	1801.0	1765.1	1772.5
1000	1854.7	1613.5	1628.6	1661.3	1643.6	1632.6	1609.5	1611.7
2000	1725.5	1493.7	1528.0	1518.6	1550.0	1489.2	1473.4	1480.1
5000	1590.8	1424.1	1466.9	1373.7	1480.6	1396.6	1348.4	1352.1
10000	1581.8	1386.9	1432.5	1306.3	1442.3	1364.8	1287.8	1288.6
20000	1577.7	1364.3	1402.1	1256.5	1422.7	1341.9	1241.6	1241.7
25000	1581.4	1358.7	1392.5	1244.2	1413.2	1335.7	1227.3	1229.0
Traveling salesman problem								
Iteration	I	II	III	IV	V	VI	VII	VIII
0	293985.9	292714.2	292920.2	293120.4	292967.8	292977.5	293108.7	292834.0
100	341619.9	288221.9	288825.4	292679.0	288479.5	290328.1	290314.7	289269.1
500	461707.2	280190.6	280566.5	285446.8	280392.8	282651.9	282931.2	281813.7
1000	563016.9	276809.3	277017.5	281450.0	276835.1	278948.1	279693.6	278341.4
2000	575346.3	274590.4	274153.3	278057.3	273991.3	275782.5	276557.0	275499.3
5000	673394.7	272519.9	272322.5	275068.4	272245.5	272920.8	273336.3	272899.9
10000	697337.3	272085.2	271452.4	273558.0	271656.8	271561.3	271830.9	271981.1
20000	683691.8	271998.7	271003.3	272645.0	271451.5	270769.1	270970.9	270972.4
50000	717214.7	271951.2	270623.8	271634.0	271406.7	270289.0	270240.2	270150.1

Those results show average values for 30 simulations. In each case the results obtained for the best individual in a given iteration are provided. Except for the selection operation all other processes are assumed to have identical parameters. The tables comprise results at different stages of evolution so as to investigate the changes in operation of the selection methods. As follows from the data presented, mixed selection is superior to other methods considered . This is due to the adequate balance between selective pressure and ability of preserving the population diversity. Both versions behave in quite similar way. It should be noted that the deterministic roulette method is fast in the initial phase of evolutionary process. However, in further stages its advantage diminishes due to the high level of unification of the population. The histogram selection is somewhat slower in the initial phase than deterministic roulette and it behaves very well in the TSP problem. The tournament selection is close to the proposed methods and has very good parameters, but is also a little bit slower (about 2-4%) than the mixed approach. The stochastic remainder method with repetitions resembles the deterministic roulette. The method of selection of best individuals is rather average. Classic roulette is very poor in all the cases presented and for

TSP (starting from not a randomly initialized population) it is even divergent.

The promising properties of mixed selection depend on the probability of its component appearance. In order to find better values of these probabilities a number of simulations have been carried out in the range $(0, 1)$ with step of 0.1. Results of simulations are presented in Table 5.

Table 5. A comparison of the mixed selection performance for various constant probabilities on different stages of the evolution.

Scheduling of time dependent jobs								
p_h	p_d	0	500	1000	2000	5000	10000	250000
1.0	0.0	2343.7	1772.2	1626.4	1487.0	1399.2	1366.3	1339.4
0.9	0.1	2337.7	1773.7	1626.4	1491.8	1375.3	1314.4	1264.5
0.8	0.2	2330.1	1759.9	1609.8	1476.9	1359.8	1303.0	1247.2
0.7	0.3	2358.6	1790.8	1634.4	1490.4	1354.6	1289.3	1230.9
0.6	0.4	2362.5	1761.6	1612.2	1475.4	1360.3	1298.6	1239.7
0.5	0.5	2334.4	1772.3	1611.6	1468.1	1343.6	1287.3	1229.3
0.4	0.6	2344.3	1759.6	1598.8	1475.5	1343.5	1281.4	1223.8
0.3	0.7	2355.6	1754.5	1600.5	1462.1	1336.8	1280.2	1225.9
0.2	0.8	2336.0	1751.2	1595.1	1468.0	1351.6	1300.1	1246.6
0.1	0.9	2338.5	1741.9	1590.8	1472.1	1363.8	1309.3	1251.4
0.0	1.0	2329.6	1755.9	1610.1	1481.5	1410.3	1379.1	1345.2
Traveling salesman problem								
p_h	p_d	0	500	1000	2000	5000	10000	250000
1.0	0.0	293022.1	277965.1	275016.6	273035.9	271311.6	270792.8	270725.1
0.9	0.1	292851.7	278039.0	275299.7	273235.3	271814.0	271131.8	270770.8
0.8	0.2	292709.4	279030.7	275583.0	273388.9	271634.6	271089.5	270729.2
0.7	0.3	292532.3	278236.7	275771.7	273884.7	272192.5	271638.5	271425.7
0.6	0.4	292526.1	278653.2	276049.3	274289.7	272505.8	271714.0	271106.4
0.5	0.5	293037.0	279175.8	276299.6	274511.1	272901.5	271964.5	271505.1
0.4	0.6	293456.6	279016.6	276748.7	274525.0	272961.3	272096.0	271217.3
0.3	0.7	294227.5	277955.7	274835.8	273062.0	271609.1	271039.0	270527.3
0.2	0.8	292850.5	276754.4	274105.8	272713.7	271666.7	271177.8	270856.0
0.1	0.9	293057.8	277903.6	275011.4	273538.0	272243.0	271964.4	271377.1
0.0	1.0	292867.9	277188.1	275059.4	273288.6	272100.4	271897.6	271870.1

One may notice that good parameters for mixed selection are located in the range $0.4 < p_h < 0.6$ ($p_d = 1 - p_h$), especially for the scheduling problem, where only blind operators were used and the method of selection plays the main role in the development of the population. It has been observed that the best values for this problem are in the vicinity of the point ($p_h = 0.45$, $p_d = 0.55$). In the case of TSP, there are several values of p_h and p_d, which give a good behavior of the algorithm. To solve the TSP problem several knowledge-based operators were used and selection is not the only one factor, which directs the evolution of the population.

4.4. The comparison of computational load for the investigated methods

The fact that evolutionary algorithms require long time of computations is widely known. In this section a short comparison of the computational load for one previously known method and the proposed methods is made. Results have been obtained on a rather slow PENTIUM-100 machine, using LINUX operating system and a program written in C++. Simulations have been performed for the scheduling problem for three cases:

- tournament selection without tuning of the operators' probabilities;
- mixed selection with constant probabilities of appearance of its component selection methods and tuning of the operators' probabilities;
- mixed selection with adaptation of probabilities of appearance of its component selection methods and tuning of the operators' probabilities.

All performed simulations (10 for each case) have lasted 10,000 iterations, minimum, maximum and average times of execution have been stored. The results are presented in Table 6.

Table 6. A comparison of execution times for selected methods.

Method	Minimum time [s]	Average time [s]	Maximal time [s]
Tournament selection without improvements	5452.5	5489.6	5600.8
Mixed selection with constant probabilities...	5479.5	5507.9	5547.9
Mixed selection with adaptation of probabilities...	5495.8	5515.3	5584.8

As it can be seen, the proposed methods do not increase the computational load seriously (by about 0.47%). Hence, the advantages of the proposed methods prevail over the increase of the computational burden.

5. Conclusions

An evolutionary algorithm is a random process and this randomness is the basis of its operation and cannot be eliminated, although it causes a rather slow operation of the algorithm. It is possible, however, to slightly direct the evolutionary computations toward betters solutions using heuristics or methods inspired by nature. This paper describes two examples of such methods. They improve (or enable) the self-control of the evolutionary algorithm.

The methods described are not limited to problems shown in this paper and may be widely used to improve the performance of the evolutionary algorithms. They ensure a speedup of the computations and a better final solution than the traditional methods, based on constant probabilities for genetic operators and simple selection methods.

Further possibility of developing the adaptive abilities of the evolutionary algorithm consists in application of principles of evolutionary programming, which gives not only the possibility of adapting the parameters of operators and selection, but also the evolutionary search for new operators, better adjusted to the specific features of the solved problem.

References

AGAPIE, A. (1999) Adaptive Genetic Algorithms – Modeling and Convergence, *Proc. of the 1999 Congress on Evolutionary Computation CEC'99*. Piscataway NJ.

ARABAS, J. (2001) *Wykłady z algorytmów ewolucyjnych* (*Lectures in evolutionary algorithms*; in Polish) WNT, Warszawa.

ARABAS, J., MICHALEWICZ, Z. and MULAWKA, J. (1994) GAVaPS – a genetic algorithm with varying population size. *Proc. of ICEC'94*. IEEE Press, 73–78.

BAKER, E. (1985) Adaptive selection methods for genetic algorithms. *Proc. of an International Conference on Genetic Algorithms and Their Applications*, 101–111.

CRONEN, B.G.W. and EIBEN, A.E. (2001) Stepwise Adaptation of Weights with Decay on Constraint Satisfaction Problem. *Proc. of the Genetic and Evolutionary Computation Conference GECCO-2001*. Morgan Kaufmann Publishers.

DAVIS, L. (1989) Adapting Operator Probabilities in Genetic Algorithms. *Proc. of the Third International Conference on Genetic Algorithms.*

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

DEJONG, K. (1975) An Analysis of the Behavior of a Class of Genetic Adaptive Systems. *Ph.D. Dissertation*. University of Michigan.

EIBEN, A.E., HINTERDING, R. and MICHALEWICZ, Z. (1999) Parameter Control in Evolutionary Algorithms. *IEEE-EC*, **3**, 2.

GOLDBERG, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

GOLDBERG, D.E., DEB, K. and KORB, B. (1991) Do not Worry, Be Messy. *Proc. of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers.

GREFENSTETTE, J.J. (1986) Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems*, Man and Cybernetics, **SMC-16**, 1.

GUNTER, R. (1999) Self-Adaptation and Global Convergence: A Counter-Example. *Proc. of the 1999 Congress on Evolutionary Computation CEC'99*, Piscataway NJ.

HOLLAND, J. (1975) *Adaptation in Natural and Artificial Systems.* Ann Arbor, The University of Michigan Press.

JULSTROM, B. A. (1995) What Have You Done for Me Lately? Adapting Operator Probabilities in a Steady-State Genetic Algorithm. *Proc. of the Sixth International Conference on Genetic Algorithms.* University of Pittsburgh.

KRINK, T. and URSEN, R.K. (2000) Parameter Control Using the Agent Based Patchwork Model. *Proc. of the 2000 Congress on Evolutionary Computation*, Piscataway NJ.

MICHALEWICZ, Z. (1996) *Genetic Algorithms + Data Structures = Evolution Program.* Springer-Verlag, Berlin Heidelberg.

SCHAEFER, R. (2002) *Podstawy genetycznej optymalizacji globalnej* (*Foundations of the genetic global optimization*; in Polish). Wydawnictwo Uniwersytetu Jagiellońskiego.

SMITH, J.E. and FOGARTY, T.C. (1997) *Operator and Parameter Adaptation in Genetic Algorithms.* Soft Computing.

SYSLO, M.M., DEO, N. and KOWALIK, J.S. (1995) *Algorytmy optymalizacji dyskretnej* (*Algorithms of discrete optimization*; in Polish). Wydawnictwo Naukowe PWN.