Raport Badawczy

Research Report

RB/22/2001

Biologically inspired methods of an evolutionary algorithm control

Jarosław 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

Pracę zgłosił: Jarosław Stańczak

Warszawa 2001

Biologically Inspired Methods of an Evolutionary Algorithm Control

Jarosław Stańczak

Instytut Badań Systemowych PAN

ul. Newelska 6, 01-447 Warszawa

stanczak@ibspan.waw.pl

Abstract

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

The second idea developed in this paper is a self-adapting method of selection. Methods applied in the evolutionary algorithms are usually derived from nature and prefer solutions where the main role plays randomness, competition and fight among individuals. In the case of evolutionary algorithms, where populations of individuals are usually small it causes a premature convergence to local minima. To avoid this drawback we propose to apply an approach based rather on an agricultural technique. The correctness of such assumption follows from the observation that by operating on small populations of plants or animals it was possible to cultivate species of desired features without randomness, fight and competition. Two new methods of object selections are proposed: a histogram selection and a

mixed selection. Also advantages of passing them into the evolutionary algorithm are shown, using examples based on scheduling and TSP.

Keywords

genetic algorithms, adaptation, adaptive evolutionary algorithms.

1. Introduction

In early years of evolutionary algorithms (EA) development much attention has been devoted to show their similarity to processes observed in nature [12]. Such concepts as a strong competition among individuals and fight for survival were introduced. Also genetic operators, which modify the genetic code of members of the population were similar to natural ones. As follows from experiments, traditional evolutionary mechanisms are not sufficient in many cases for fast growing of population in desired direction. More complicated, adaptive methods are much better. There are big difficulties with a theoretic description of its behavior and properties, but there are first results, some of them very surprising [11], [1].

In the classic genetic algorithm, proposed by Holland [12], both used operators 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 pass a high probability (0.8 - 1.0) for crossover and a low one for mutation (0.01 - 0.2). These values were chosen not only intuitively, but based on biological and experimental data. The mutation is responsible for exploration of searched domain, while the crossover exploits previously found best regions. Too big level of mutation can lead to lose of convergence to optimal solution, because of the fact that changes of the genetic material are stronger than directed by fitness function process of evolution. So the probability of it is very low. High

level of crossover intensifies exploration of local extrema and it is useful to have big probability of it.

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 designed for operating on them genetic operators. 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 an actual operator. So we have two solutions to overcome this problem: experimental tuning of probabilities or to make them self-adaptive[1]. There were some trials of experimental evaluating parameters of genetic optimization for several problems, using traditional genetic algorithm [6], but it is rather impossible to investigate the whole domain of possible modifications of genetic algorithms for different problems. Of course tuning parameters for actual problem is still a very good (but time consuming) way of finding optimal values of them. The second possibility is much more promising. One of the earliest solutions was to use a genetic subalgorithm to optimize parameters of main genetic optimization process [10]. But it was rather slow method. Next step was to find the probability of an appearance of the genetic operator connected with his behavior. Such methods were described in [3], [4], [5], [13]. They are based on a quality of the operator which modifies the population of solutions. Every operator gets (if is chosen) his period of time when he affects the population. So all modifications of an evaluation function can be assigned to a particular operator and they modify his probability of appearance. They are also backpropagated to previously used operators, which credits are also taken into account and their probabilities are also changed. Thanks to this method we can use any number of operations, no matter how they work, randomly or knowledge based. The method described in the chapter 2 of that work continues and develops the latter mentioned idea.

---

[1] A great survey of almost all used methods of the parameter control in EA, can be find in [7] or [16].

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

2. A new method of probability control

2.1 Description of the method

In the presented approach it is assumed that an operator which generated good results should have bigger probability and more frequently effect the population. But it is very likely that operator, which is good for one individual, gives worse effects for another, for instance because of his location in the domain of possible solutions. So every individual may have his own preferences[2]. It is rather common situation in nature - every living creature has his own needs concerning environment of live, food, temperature, light, etc. Population of solutions created for solving technical problems has also biological origins and probably can develop

[2] The idea of „personal preferences" of population member is also used in [14] but quite different method of adaptation.

better, considering preferences of its members. The idea of personal preferences is realized as follows. Every individual has a vector of floating point numbers - q (beside 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 is, the higher is the probability of the operator. This relationship may be written as follows:

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

where:

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

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

$L(t)$ - 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 starting values of q_0;
2. i := 1;
3. Selection of genetic operators (and partners) by 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;
7. Selection of members to the offspring population;
8. i := i + 1;
9. If not stop then go to 3;
10. End.

This method can be applied both in the case of operators changing one individual (like mutation) and two or more individuals (like crossover). In the first case, there is no problem in executing operators, it depends only on personal preferences. In the second one, two (or more) individuals must choose the same operator to make it executable. So it is necessary to

implement a mechanism of searching for population members with the same preferences and this solution is very similar to situations from nature.

The parameters q_{ij} are closely related with values of the fitness function or to be exact with changes of this function worked out by the genetic operators. It is possible to attribute every change of fitness function to an actual operator, because only one operator modifies one member of population in one generation. Of course during one generation different operators are used by different individuals, but one individual is changed only by the chosen operator. The formula (2) shows how the qualities of operators are estimated.

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

where:

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

l - number of the chosen operator;

q_0 - a credit value, always significantly less than 1;

$f(..)$ - normalizing function, its arguments can be:

$Q(t-1)$ - the best solution in the moment $t-1$;

$\bar{x}_j(t)$ - the mean value of improvements of quality function;

$x_{ji}(t)$ - 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 zero, $x_{ji}(t)=Q(t-1) - Q_{ij}(t)$ (minimization) or $x_{ji}(t)=Q_{ij}(t) - Q(t-1)$ (maximization), $Q_{ij}(t)$ - solution of *j-th* individual, obtained using *l-th* operator;

α - a coefficient of forgetting (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 corresponding to it operation for current individual and for its possible offspring. This fact is not profitable, because it is possible that the excluded operator will work better on other stages of the evolution process. For exploring operators like mutation it is often necessary to let them work even without any visible improvements of the fitness function.

The second addend is 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 global best solution (which is more desirable) and improvement of the offspring in relation to its parents. Both possibilities can be used in the formula (2) with appropriate weights higher for the first and lower for the second way. The normalization function is responsible for making changes of quality function independent on character of the problem. It is easy to imagine problems where a small change of quality problem is for instance 10^5 or 10^{-5}. This situation would require a long process of tuning parameters of formula (2), if the normalization function wasn't applied. After adding normalization the range of searched optimal values of parameters q_0 and α can be significantly decreased. Exact form of normalized function is not given in formula (2). It is possible to use any function which transforms values of quality function improvements to range (0, 1) and higher positive changes are also higher after normalization. Successfully tested by authors functions are: best value of quality function reached by optimization process and a 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 remembering old achievements of an operator multiplied by a forgetting factor α. The parameter α is responsible for balancing the influence of the quality factor of an operator old and new improvements. It should be noticed that some genetic operators may achieve good results in some phase of simulation,

and then draw out their abilities. In the other hand remaining operators, probably better in next phases, would have small probabilities of appearance, if the factor α hasn't been introduced. So it would take a lot of time to change this situation and slow down the process of genetic computations. The effect of forgetting former achievements can overcome this problem. When operators don't change the global best solution for some time, the probabilities of operators become small or even equal. After every generation only bounded with the chosen operator value q_{ij} 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 others, not selected operators. Setting the value of α to 1 can cause the situation of domination of operators which don't produce any income, but were good in early stages of evolutionary process and consequently slew down computations. In other hand value 0 causes the situation where quality of operators bases only on the newest 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 time. In the case of long term lack of positive achievements of an operator, its value q_{ij} establishes at level described by the formula (3):

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

where:

$q_{ij}(\infty)$ - a limit value of $q_{ij}(t)$, all other symbols like in formula (2).

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

2.2 Parameters for the practical use

The formula (2) requires two numerical parameters q_0, α and a normalizing function. Process of selecting this elements is not very complicated and is described below. Generally, the parameter q_0 ought to be less than important changes of normalized quality function. The parameter α should belong to the range (0, 1) to assure a convergence of geometric sequence, generated by formula (2). The span of practically used values of α is narrower (0.7, 1). Too small values of α cause lack of positive feedback between a high quality of operator and its frequency of appearance, because all qualities fast become equal for all operators. Too big values cause in practical eliminating of worse operators at the beginning of the evolution though they can become very useful later. Even selecting $\alpha=1$ shouldn't 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 of setting initial values of q_{ij} is using 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.

The formula (3) may be also used as a base for determination values of α and q_0. It is possible to accept an assumption that the quality factor of an operator can change its value in the first iteration maximally in 1.0 (using normalizing function as in formula (8), for functions (6) and (7) this value depends on the solved problem):

$$\frac{x_{ij}(1)}{f(Q(0),\bar{x}_j(1))} \leq 1{,}0 \tag{4}$$

It should be an important change in relation to the coefficient $q_{ij}(\infty)$ in for instance 50%, so it formulates a following estimation:

$$\frac{q_o}{1-\alpha} * 0{,}5 = 1{,}0 \tag{5}$$

where:

all symbols in (4) i (5) have the same meaning as in (2) i (3).

Assuming value $\alpha = 0.9$[3] it is possible to determine a value of q_0 as 0.2 and $q_{ij}(\infty) = q_{ij}(0) = 2$. Described way of finding parameters of formula (2) is only a scheme but gives satisfying values. Generally, the method of adaptive selection of operators is resistant on selected values of parameters. They can be selected from a wide range without any malicious effects because the main role of valuing operators play their achievements.

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

- in the case of negative values of quality function it is necessary to use a modulus of the function:

$$f(Q(t), \bar{x}_j(t)) = |Q(t)| \tag{6}$$

where:

$Q(t-1)$ - the best solution in the moment $t-1$;

$\bar{x}_j(t)$ - the mean value of improvements of quality function;

[3] This value is chosen, basing on conducted experiments with various values of the parameter α.

- in the case of approaching the quality function to zero, its normalized value could unlimitedly increase due to division by zero. Scaling and translation of the quality function can overcome this problem (parameters a and b used in (7) are strictly connected with solved problem):

$$f(Q(t),\bar{x}_j(t)) = b + a \cdot Q(t) \tag{7}$$

where:

$Q(t-1)$ - the best solution in the moment $t-1$;

$\bar{x}_j(t)$ - the mean value of improvements of quality function;

a, b - chosen coefficients;

- in the situation of changing sign of the quality function, described above method can fail. In that case a good solution is using a mean value of improvements of the quality function, computed separately for every individual of the population:

$$f(Q(t),\bar{x}_j(t)) = \bar{x}_j(t) = \frac{\bar{x}_j(t-1)\cdot(t-1) + x_{ij}(t)}{t} \tag{8}$$

where:

$Q(t-1)$ - the best solution in the moment $t-1$;

$\bar{x}_j(t)$ - the mean value of improvements of quality function;

The latter case seem to be the most interesting. For this normalizing function is easy to foresee the maximum value of increase of the quality factor, used in formula (4). The second advantage of this function is adaptation of the mean value of improvements, because it decreases during computation and also possible improvements decrease on more advanced states of evolution. This effect compensates decreasing level of evolution during computations.

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

3. Uncontrolled and controlled selection methods

3.1 A short survey of traditional selection methods

In traditional evolutionary algorithms the following methods of not controlled selection are used [15], [8]:

- The roulette method. Individuals to the offspring population are selected in accordance with probabilities, which are proportional to the values of their matching function. An expected value of offspring population can be written as:

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

where:

$E(t+1)$ - 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 matching function for *l-th* or *j-th* individual in iteration t.

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

$$n_l(t+1) = \mathbf{1}(F_l(t) - F_{\min}(t)) \tag{10}$$

where:

$\mathbf{1}(...)$- step function, $\mathbf{1}(x) = \begin{cases} 0 \text{ for } x \le 0 \\ 1 \text{ for } x > 0 \end{cases}$;

$F_{min}(t)$ - all individuals evaluated above this value, enter to the next population;

all other symbols are the same as in the formula (9).

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

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

where:

$n_l(t+1)$ - number of offspring population of l-th individual in the next generation;

$f(..)$ - an function converting a real value to an integer value (*round* or *floor*);

μ, λ, $F_l(t)$, $F_j(t)$ - have the same meaning like in the formula (9).

- Selection by stochastic remainder method with repetitions is some kind of combined method. Its first phase is based on deterministic roulette method (11). Free places after first part are completed using fractional parts of „individuals" by traditional roulette method, it is described by the formula (12).

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

where:

x_l , x_j - an expected value of offspring in the roulette selection (9), denoted as x_l or x_j for shortening the formula (12);

all other symbols like in formula (9) and (11).

- Tournament selection is a method, where μ times a competition between two (sometimes more) randomly chosen individuals is conducted and a better one is connected to the

descendant population. The expected value of the *l-th* individual offspring shows the formula (13):

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

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, form which the best is a winner).

This list doesn't exhaust the whole domain of different selection methods; also very interesting ideas cover ranking selection [2], [9], and many others. Traditional roulette approach in stationary evolutionary algorithm provides usually poor results. After long time of computations it stabilize oscillating round about some value far off the global extremum. Poor properties of the roulette method were criticized in very early works on genetic algorithms [6]. Much better results may be achieved using the selection of best individuals. However, this method doesn't save a high level of pressure for the population development. The deterministic roulette method has a very strong pressing 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 traditional methods of selection it has been found that they may not be controlled and adapted. Thus, they do not enable to influence the process of selection and

development of population by some external factors. Presented in this paper histogram and mixed selection methods allow fulfilling this postulate.

3.2 A 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 matching function is created (this list resembles a histogram).

$$n_i(t+1) = f\left(\mu \cdot \frac{F_i(t)}{\sum_{j=1}^{j=s} F_j(t)}\right) \tag{14}$$

where:

s - number of values on the list;

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

The length of this list is usually shorter than a number of individuals in the population. Next, a mean value of the matching function is calculated but 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) passes to the offspring population adequate number of individuals. First, the particular matching 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 case when 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 opposite case some the worst individuals are removed. Fluctuations of obtained size of base population are caused by approximations calculated real values by integers.

The method of histogram selection delivers a number of interesting features. This selection is carried out in deterministic way, so there is no possibility of dying out the best individuals, which can occasionally happen in case of probability based methods. It also enables to maintain the population diversity in a simple manner. In this approach mean value of the matching function depends only on existing in the population values of this function, not on number of appearances of these values in the population. Thus, the best individuals are not advanced excessively. Worse individuals also have got a chance to be selected to new population. However, there is a possibility to lose good individuals described by the same matching function values but with completely different genetic code. So the histogram selection may be extended to distinguish solutions having the same values of matching function but representing different individuals. In 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 similarity of individuals. In this case one should introduce some neighborhood of respective point in which two solutions are considered as identical.

3.2.2 Mixed selection

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

Iteration	μ	λ	μ	λ
	Histogram Selection		**Deterministic Roulette Method**	
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 Method		**Histogram Selection**	
4	5, 5, 4, 4, 3		5, 5, 4, 3, 2	

Table. 1. An example of the mixed selection performance.

In the table 1 an example of a certain matching function maximization is considered. Two possible cases for randomly chosen selection methods are provided. Particular fields of the table comprise values of the matching function where each number suits to one individual. To simplify and better illustrate advantages as well as faults of two selection methods a lack of improvement in this small part of the algorithm was assumed. Such situation occurs very often when computations last. In the respective example the population size was $\mu=\lambda=5$ for the strategy $(\mu+\lambda)$. The above hypothetical example of evolutionary computations shows characteristic properties of the proposed method. The deterministic roulette selection exhibits tendency to fill up the population by identical individuals, on the other hand the histogram selection assures to preserve the population diversity. As follows from this example, the histogram selection also has the property to repair a composition of too uniform population. Both versions of selection (histogram, deterministic roulette) supplement each other. It is possible to control the property of selection operation by introducing the probability of appearance of particular selection version. The more frequent appearance of the roulette selection causes the higher pressure towards promoting 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 the population diversity, paying for it the

weaker pressure towards promoting the best individuals. Due to this mechanism it is possible to examine given search area more effectively.

This approach allows adjusting features of the selection operation to current requirements of the population. The adaptation of probabilities of selection operations may be derived from statistical properties of the population. An idea of this method presents Fig. 1 (in equations $p_{his}=1-p_{det}$).

1. If $3*\sigma(F) < max(F_{mean} - F_{min}, F_{max} - F_{mean})$ then $p_{his} = p_{his} - a*p_{det}$;
2. If $0,5*\sigma(F) > max(F_{mean} - F_{min}, F_{max} - F_{mean})$ then $p_{his} = p_{his}*(1 + a)$;
3. If $0,5*\sigma(F) \leq max(F_{mean} - F_{min}, F_{max} - F_{mean}) \leq 3*s(F)$ then $p_{his} = p_{his}*(0,5-p_{det})*a$;

Fig. 1. The algorithm of automatic tuning probabilities of the mixed selection elements.

The symbols used in Fig. 1 mean: p_{his} - probability of histogram selection appearance, p_{det} - probability of deterministic roulette, F_{mean}, F_{min}, F_{max} - average, minimal and maximal values of fitness function in the population.

If particular individuals are described by too small standard deviation of the matching 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 can be 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. In our experiments the statistic parameters of the population: 0.5 and 3.0 have been found empirically because they proved the best results. The value of the parameter a has been fixed to 0.05. The method of tuning parameters of the selection operation also has been practically checked. It has been found that this approach is better than the selection with constant (but best found) value of a.

4. Experimental results

4.1 Solved problems

To assess the utility 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: blind and heuristic. All operators were designed to assess problems limitations - the offspring always encoded valid solutions. The operators were:

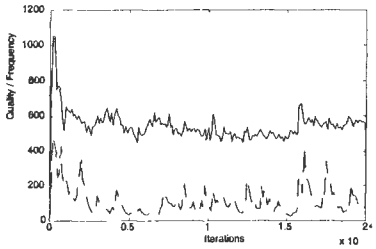
- mutation - a random exchange of two cities in the list of cities.
- crossover - starting from the first city of one list, next cites are chosen from one or second list to be closer to the previously accepted.
- inversion - a fragment of the list is used in the reverse order;
- transposition - a fragment of the list of cities is moved to another part of the list;
- two-optimal method - exchanges two chosen edges in the route, if it gives a shorter route;
- 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 randomly chosen one. The „neighborhood-2" operator cuts a close to chosen city from the path, moves all cities between them in one position an inserts the city near the chosen one.

Members of the population for scheduling were coded as lists of tasks waiting for processing. Also a graph of time-dependencies was introduced into algorithm as labels of

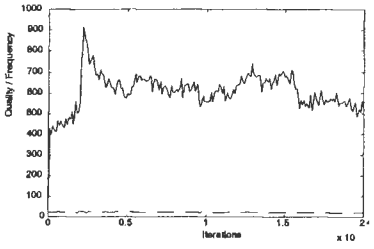
waiting tasks. All operators are designed to assess valid solutions. Only problem-blind operators were used (similar to used for TSP): mutation, inversion and transposition.

4.2 The probability control

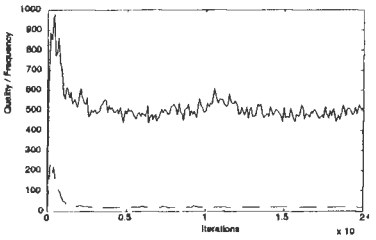
Fig. 2 shows the dependence between a quality of an operator and its frequency of an appearance for selected operators (only TSP). All data are collected from the whole population (not from one member) so that's why there is no exact proportion between quality and frequency (fig. 2b shows a very big jump of frequency and very small increase of quality - it can be caused by an important improvement of fitness function of one member, which had many „children"). It would be difficult to show data from one individual during the iterations, because it lives only for one epoch.



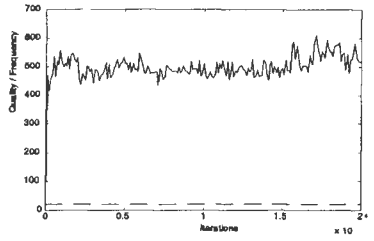
a) crossover



b) „neighborhood-2"



c) two-optimal



d) „neighborhood-1"

Fig. 2. The dependence between a quality factor of an operator (at the bottom of every picture) and its frequency of appearance (TSP - 1002 cities).

Analyzing those pictures clearly can be seen that better behavior of the operator effects in an increase of frequency of his appearance and consequently speeds up genetic computations (which can be seen from tables 2 and 3). Operators which don't bring any income aren't eliminated, but have approximately constant frequency. It is very likely that they behave in the way like mutation and have exploration properties. They are very important, but they can't appear too often, due to possible effect of loosing convergence to optimal solution.

In the table 2 an influence of the factor α on the genetic computations ca be observed. Data from 25 simulations are collected and a mean values of them are presented for different numbers of iterations and values of factor α.

	0	100	500	1000	2000	5000	10000	20000	50000
const	293042	289893	282191	278390	275278	272749	271492	270593	269693
α=0.0	293246	289378	280440	277136	274487	272270	271399	270607	269786
α=0.1	293351	287947	279570	276632	274347	272323	271298	270803	269929
α=0.2	293395	288355	279539	275973	273609	271842	271021	270527	269855
α=0.3	293258	288106	280058	276872	274388	272401	271577	270975	270434
α=0.4	293026	287169	279246	276007	273873	271972	271199	270591	269728
α=0.5	292938	287472	278819	275744	273735	271895	271182	270759	270068
α=0.6	293635	288056	278282	275834	273875	272067	271422	270998	270154
α=0.7	292542	286664	278404	275954	273981	272269	271659	271320	270788
α=0.8	292832	286319	277131	275368	273503	271981	271506	271286	270492
α=0.9	293522	286268	276100	274115	272655	271345	270937	270560	269942
α=1.0	292957	285490	275853	275602	274751	272768	271235	270858	270359

Table 2. A comparison of data obtained from 25 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.

All simulations were started from solutions obtained from a simple algorithm that generates a route taking the closest city to the last visited. Starting from randomly chosen cities it gives different solutions and works much faster than random initialization of the solution population. A strategy $(\mu+\lambda)$ was used with $\mu=\lambda=60$. The optimal solution for the

considered task is known and equals 259045. The best found solution by described program is 3% worse after 10^6 epochs.

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

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

Considering data from table 2 and table 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 $\alpha=0$ doesn't provide equal values of probabilities for all operators because it doesn't eliminate the influence of their behavior. It only provides the smallest period of remembering their achievements, which lasts till the next execution of that operator.

Parameters of simulation for scheduling are the same as in TSP. Optimal solution is unknown, best found 1121 (after 10^6 epochs). Optimal values of α are from the same range, but differences between solutions for various α are bigger than in the case of TSP. This fact is connected with used operators. For TSP both blind and knowledge-based were used and they played the main role in computations. In scheduling only randomly working

operators generated improvements and proper 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 provided for two, described earlier 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. Results of simulations are provided in Table 4.

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

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_{his}=0,45 and p_{det}=0,55), VIII- mixed selection with automatic tuning of probabilities.

That results show average values for 30 simulations. In each case results obtained for the best individual in given iteration are provided. Except the selection operation all

other processes are assumed to have identical parameters. The tables comprise results at different stages of evolution to investigate changes in operation of the selection methods. As follows from the presented data the mixed selection is superior to the other considered methods. It is due to adequate balance between selecting pressure and ability of saving the population diversity. Both versions behave in enough similar way. It should be noted that deterministic roulette method is fast in initial phase of evolutionary process. However in further stages its advantage is diminished due to too big unification of the population. The histogram selection is somewhat slower in initial phase than deterministic roulette and it behaves very well in the TSP problem.

The tournament selection is close to proposed methods and has very good parameters, but also is a little bit slower (about 2-4%) than mixed. The stochastic remainder method with repetitions resembles deterministic roulette. Selection of best individuals method is rather average. Classic roulette is very poor in all presented cases and it is widely known that its importance is limited rather for theoretic considerations than practical use.

Promising properties of the mixed selection depend on the probability of its components appearance. To find better values of these probabilities a number of simulations have been carried out in range (0, 1) with step 0.1. Results of simulations are presented in the table 5. One may notice that good parameters for the mixed selection are located in the range $0.4<p_{his}<0.6$ ($p_{det}=1-p_{his}$) especially for scheduling problem, where only blind operators were used and a method of selection plays the main role in development of population. It has been observed that the best values for this problem are in vicinity of the point ($p_{his}=0,45$, $p_{det}=0,55$). In the case of TSP, there are several values of p_{his} and p_{det}, which assure a good behave of algorithm. To solve this problem several

knowledge based operators were used and selection is not the only one factor, which directs the population.

Scheduling of time dependent jobs								
P_{bls}	p_{det}	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_{bls}	p_{det}	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

Table 5. A comparison of the mixed selection performance for different probabilities

5. Conclusions

Described methods, which improve the controlling of the evolutionary algorithm are not limited only to problems shown in this paper and may be widely used for optimization the performance of the evolutionary algorithm. They assure a speedup of the computations and a better final solution than using traditional methods, based on constant probabilities for genetic operators.

Further possibility of development of described methods is by applying principles of evolutionary programming, which gives not only the possibility of adapting parameters

of operators, but also evolutionary searching for new operators, better adjusted to the specific of solved problem.

REFERENCES

[1] A. Agapie, *Adaptive Genetic Algorithms - Modeling and Convergence*, Proceedings of the 1999 Congress on Evolutionary Computation CEC'99, Piscateway NJ, 1999.

[2] E. Baker, *Adaptive selection methods for genetic algorithms*, Proceedings of an International Conference on Genetic Algorithms and Their Applications, 1985.

[3] B. G. W. Cronen, A. E. Eiben, *Stepwise Adaptation of Weights with Decay on Constraint Satisfaction Problem*, Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001, Morgan Kaufmann Publishers, Las Vegas, Nevada, USA, 2001.

[4] L. Davis, *Adapting Operator Probabilities in Genetic Algorithms*, Proceedings of the Third International Conference on Genetic Algorithms, 1989.

[5] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[6] K. DeJong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Ph.D. Dissertation, University of Michigan, 1975.

[7] A. E. Eiben, R. Hinterding, Z. Michalewicz, *Parameter Control in Evolutionary Algorithms*, IEEE-EC, Vol. 3, No. 2, 1999.

[8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[9] D. E. Goldberg, K. Deb, B. Korb, *Do not Worry, Be Messy*, Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo CA, 1991.

[10] J. J. Grefenstette, *Optimization of Control Parameters for Genetic Algorithms*, IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-16, No. 1 January/February 1986.

[11] R. Gunter, *Self-Adaptation and Global Convergence: A Counter-Example*, Proceedings of the 1999 Congress on Evolutionary Computation CEC'99, Piscateway NJ, 1999.

[12] J. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, The University of Michigan Press, 1975.

[13] B. A. Julstrom, *What Have You Done for Me Lately? Adapting Operator Probabilities in a Steady-State Genetic Algorithm*, Proceedings of the Sixth International Conference on Genetic Algorithms, University of Pittsburgh, 1995.

[14] T. Krink, R. K. Ursen, *Parameter Control Using the Agent Based Patchwork Model*, Proceedings of the 2000 Congress on Evolutionary Computation, Piscateway NJ, 2000.

[15] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Program*, Springer-Verlag Berlin Heidelberg, 1996.

[16] J. E. Smith, T. C. Fogarty, *Operator and parameter adaptation in genetic algorithms*, Soft Computing, June 1997.