Computer Science • 25(1) 2024 https://doi.org/10.7494/csci.2024.25.1.5330

Jarosław Stańczak

EFFICIENT SELECTION METHODS IN EVOLUTIONARY ALGORITHMS

Abstract *Evolutionary algorithms mimic some elements of the theory of evolution. The survival of individuals and the ability to produce offspring play significant roles in the process of natural evolution. This process is called natural selection. This mechanism is responsible for eliminating weaker members of the population and provides the opportunity for the development of stronger individuals. The evolutionary algorithm, an instance of evolution in the computer environment, also requires a selection method – a computerized version of natural selection. Widely used standard selection methods applied in evolutionary algorithms are usually derived from nature and prefer competition, randomness, and some kind of "fight" among individuals. But the computer environment is quite different from nature. Computer populations of individuals are typically small, making them susceptible to premature convergence towards local extremes. To mitigate this drawback, computer selection methods must incorporate features distinct from those of natural selection. In the computer selection methods randomness, fight, and competition should be controlled or influenced to operate to the desired extent. This work proposes several new methods of individual selection, including various forms of mixed selection, interval selection, and taboo selection. The advantages of incorporating them into the evolutionary algorithm are also demonstrated, using examples based on searching for the maximum α-clique problem and traditional Traveling Salesman Problem (TSP) in comparison with traditionally considered highly efficient tournament selection, deemed ineffective proportional (roulette) selection, and other classical methods.*

Keywords evolutionary algorithms, selection methods, adaptive evolutionary algorithms

Citation Computer Science 25(1) 2024: 95–122

1. Introduction

In the evolution of living creatures, the process of natural selection is the primary force guiding gradual changes in their genomes across consecutive generations and contributing to their development. The process of natural selection is not a single mechanism, it consists of several different elements overlapping each other, with some parts being deterministic and others random. The superposition of these factors affects the success of the individual. The sum of these successes of individuals gives a higher level of development of the total population and species. Hence, the selection can be compared to the strainer filtering out the weakest organisms (in any case the most likely eliminates individuals weaker than the better), thereby enabling the survival of the best features and the accumulation of these traits in subsequent generations.

In a natural environment, it is difficult to say to what extent the content of the genetic code of the individual determines the success of the development. Probably in simple organisms, it matters the most, in higher organisms, which have the ability to learn, the direct influence of their genes is smaller, but of course, the learning ability in some way comes from what is written in the DNA of the organism. Therefore, one can observe the phenomenon in which the future of individuals and species is influenced not only by genetic predisposition but also the skills acquired during individuals' lifetimes. Unfortunately, this feature of living organisms is still weakly used by evolutionary algorithms, although this is certainly a very promising opportunity, developed in memetic algorithms [12] and agent systems. Certainly, the combination of these features in the memetic or evolutionary–agent systems has a great future.

During the early development of genetic or evolutionary algorithms, much attention was devoted to mimicking processes observed in natural evolution [11]. However, applying pure natural selection in computer environment is not possible. Concepts such as strong competition among individuals, randomness, and the fight for survival have given rise to several traditional selection methods used in evolutionary computations (proportional or roulette wheel selection, tournament selection).

However, such an approach, in the case of rather small populations of individuals with which users are dealing in evolutionary algorithms, is often inefficient and leads to their premature convergence to local extremes or do not use the potential inherent in the method of evolution, significantly slowing down the computations. Depending on the method of selection, the algorithm easily falls into either of two disadvantages: the best individuals dominate the entire population and a significant slowdown or blockage of the evolution due to the small diversity of the population (too high selection pressure) or on the other side the exploration of new areas is strong, but promising solutions live too short to find near-optimal solutions (the selection pressure is too weak).

Thus, it can be seen that the selection in the computer environment must operate quite differently than natural selection [17]. Since a universal computer selection

method cannot be found, there is certainly a need to choose a method from existing ones or even develop a method, depending on the specifics of the problem being solved. Furthermore, some researchers consider the selection as one of the genetic operators. Unless there is no doubt that the set of genetic operators should be tailored to the specifics of the solved problem, spreading this idea concerning selection is usually challanging.

Accordingly, the most frequently used are quite typical and basic methods, while it would be possible to obtain much better results using a little more sophisticated methods: adaptive or a method in which the selection is disguised as another mechanism, e.g., the lifetime of an individual [2], etc.

This paper is a continuation of research, presented in [17] with several new selection methods developed from that time. Although there are some theoretical methods, that describe properties of selection methods (section 2), empirical experiments which show their behavior are also very important (section 5). Sections 3 and 4 provide detailed descriptions of the discussed selection methods.

2. Properties of selection methods

Selection in evolutionary algorithms is characterized by the concept of selective pressure. It is difficult to strictly define this notion. Mostly, it is described using coefficients, the values of which were estimated for some selection methods. However, these factors never fully reflect the nature of the method. So far, all assessments of the suitability of selection methods have been verified experimentally to observe their performance.

This is due to the absence of an adequate mathematical framework that could explain the theory and methods of operation of evolutionary algorithms, showing how the selection method affects the convergence of the algorithm to the optimum or sub-optimum.

In addition, evolutionary selection is not the only force targeting calculations for more sophisticated evolutionary algorithms. Similar or even greater importance may lie in specialized genetic operators enriched with the knowledge of the task to be solved or adapted from simple methods of local optimization, often used in advanced evolutionary algorithms and memetic algorithms.

Goldberg and Deb [10] proposed a measure of selective pressure called the *takeover time*, representing the number of generations τ needed to fill the entire population of solutions with the best copies of the same individual in the absence of modification by genetic operators (assumed to preserve a copy of the best individual to prevent accidental extinction). Unfortunately, for many of the more complicated selection methods, estimating this time analytically is challenging. Even if it could be calculated, it may be difficult to precisely assess the properties of the particular selection method looking only at its value of τ.

Another coefficient measuring selection characteristics is called *selection intensity* – I [13], defined as follows:

$$\mu_{t+1} = \mu_t + I \cdot \sigma_t \tag{1}$$

where:

μ_t, μ_{t+1} – mean values of the population fitness function values before and after selection,

σ_t – the standard deviation of the population fitness function before selection.

This factor was defined for the theoretical study aiming to assess the convergence of selection methods.

Another method for assessing selection is the concept of *genetic drift*, as presented for instance, in [15]. Genetic drift is a phenomenon observed in evolutionary algorithms, which depends on changing frequencies of genes in the population, consequently leading to the convergence of the population to identical solutions. Genetic drift is defined as follows:

$$r = \frac{E(\sigma')}{\sigma} \tag{2}$$

where:

r – the genetic drift coefficient,

$E(...)$ – a symbol of the expected value,

σ' – the standard deviation of the population fitness function after selection, and σ before selection.

The *selection variance* [5] is a factor easier to determine than the genetic drift in practical computer simulations, describing properties of selection in evolutionary algorithms:

$$V = \frac{\sigma'^2}{\sigma^2} \tag{3}$$

where:

V – the selection variance coefficient,

$E(...)$ – a symbol of the expected value,

σ' – the standard deviation of the population fitness function after selection,

σ – the standard deviation before selection.

The selection of individuals typically results in the *loss of diversity* within the descendant population. This is rather a harmful phenomenon in relatively small computer populations, especially big levels of loss of population diversity. The loss of diversity factor can also serve as a measure of selection properties [5, 6]:

$$p_d = \frac{N - |P(t) \cap P(t+1)|}{N} \tag{4}$$

where:

p_d – the measure of loss of diversity,

N – the cardinality of the population,

$P(t)$, $P(t+1)$ – populations before and after selection.

Loss of diversity in the population as a result of selection action can be minimized using non-standard methods especially designed to obtain that aim. Several of them are presented in the section 3.1. Additionally, a simple parameter indicating the level of the *population diversity* can be introduced. It is the number of different solutions in the population in proportion to the population cardinality – before and after selection. The notion "different solutions" means different in encoded solutions, not only different in the values of the fitness function, because the same values of the fitness function may characterize completely different solutions. The similarity factor in this case is a difference in at least one encoded position[1] (one bit, one city, one graph node, ..., for the real-number problem it can be some vicinity of a given point). Therefore, it is possible to analyze population diversity before and after selection:

$$s_b = \frac{N_{db}}{N_b} \tag{5}$$

$$s_a = \frac{N_{da}}{N_a} \tag{6}$$

where:

s_b, s_a – measures of the population diversity before and after selection,
N_b, N_a – the cardinality of the population before and after selection,
N_{db}, N_{da} – numbers of different solutions (encoding different points in the problem's space) in population before and after selection.

3. Standard selection methods

While numerous selection methods have been invented and can be considered standard, two of them stand out as particularly important: the proportional or roulette wheel method (the first to be used and theoretically analyzed) and tournament selection (known for its excellent practical properties and frequently employed in theoretical research). However, it's worth noting that the remaining methods are also applied in practice and, at times, in theory.

3.1. The proportional or roulette selection

It is one of the oldest selection methods used in genetic algorithms. Every individual in the parent population can have an offspring with a probability proportional to the value of its fitness function. In other words, the probability of selecting each individual is equal to the ratio of the value of its fitness function to the sum of the fitness values for the entire population. Consequently, the probability of selecting the individual and the expected value of descendants for that individual can be expressed using the following formulas:

$$p_l(t+1) = \frac{F_l(t)}{\sum_{j=1}^{\mu+\lambda} F_j(t)} \tag{7}$$

[1]Of course, it is possible to use a stronger similarity criterion with more than one difference in the encoded solution.

$$En_l(t+1) = \frac{\mu * F_l(t)}{\sum_{j=1}^{\mu+\lambda} F_j(t)} \tag{8}$$

where:

$p_l(t+1)$ – the probability of selection of the l-th individual to the descendant population $l \in 1, ..., \mu + \lambda$,

$En_l(t+1)$ – the expected value of descendants of the l-th individual, $l \in 1, ..., \mu + \lambda$,

μ – cardinality of the parent population,

λ – cardinality of the descendant population,

$F_l(t)$, $F_j(t)$ – values of fitness functions for the l-th (j-th) individual.

As it has been stated in some works, this method of selection is proper only for theoretical purposes. The real application of it is rather useless because there are better methods, especially the tournament selection [23].

3.2. The deterministic roulette (or proportional) selection method

The deterministic roulette method is a modified proportional selection in which randomness has been eliminated[2]. Additionally, some scaling skills have been introduced to improve obtaining integer and proper numbers of offspring individuals (the number of offspring should be equal or close to μ).

An individual from the parent population obtains a number of offspring by rounding the ratio of the value of its fitness function to the average value of the fitness function for the entire population to the nearest integer (or integer part), multiplied by the population's cardinality (9). The resulting values are scaled again using a similar ratio to minimize errors in the approximation of offspring cardinality. If there is depletion despite the occurrence, it is populated with the best individuals that have not entered the population or those with the highest discard fractions. The excess is disposed of by removing the appropriate number of the worst-selected individuals.

$$n_l(t+1) = f\left(\frac{\frac{(\mu+\lambda)\cdot F_l(t)}{\sum_{j=1}^{\mu+\lambda} F_j(t)} \cdot \mu}{\sum_{i=1}^{\mu+\lambda} f(\frac{F_i(t)\cdot(\mu+\lambda)}{\sum_{k=1}^{\mu+\lambda} F_k(t)})}\right) \tag{9}$$

where:

$n_l(t+1)$ – the number of offspring of the l-th individual in the descendant population, where $l \in 1, ..., \mu + \lambda$,

$f(...)$ – a function that converts the actual floating point result to integer value,

μ – the cardinality of the parent population,

λ – the cardinality of the descendant population,

$F_l(t)$, $F_j(t)$ – values of fitness functions for the l-th (j-th) individual.

[2]This method is very similar to the *selection by stochastic remainder method with repetitions* where the remaining vacant fractional parts of individuals in the population are supplemented according to the proportional selection.

3.3. The tournament selection

This selection method operates as follows. First, a random collection of a few individuals (greater than one, less than the total population number) is selected, then one of them with the best value of fitness function wins. Individuals are drawn to the tournament among the entire population with equal probability. The most commonly used is a variant in which the two individuals are drawn and to the descendant population the best one is selected, but in the general case, the size of the tournament can be any number (not bigger than the population size), and the best individual from the selected ones is copied to the descendant population. The draw is carried out as many times as there are free places for individuals in the descendant population. The probability of selecting an individual in this method is expressed by formula (10), and the expected number of individuals by formula (11) (based on [3]):

$$p_l(t+1) = \frac{(\mu+\lambda-l+1)^k - (\mu+\lambda-l)^k}{(\mu+\lambda)^k} \tag{10}$$

$$E(n_l(t+1)) = \mu \cdot \frac{(\mu+\lambda-l+1)^k - (\mu+\lambda-l)^k}{(\mu+\lambda)^k} \tag{11}$$

where:

$p_l(t+1)$ – the probability of choosing the l-th individual to the new parent population, where $l \in 1, ..., \mu+\lambda$,
$E(n_l(t+1))$ – the expected number of copies of the l-th solution in the descendant population, where $l \in 1, ..., \mu+\lambda$,
μ – the cardinality of the parent population,
λ – the cardinality of the descendant generation,
k – the size of the tournament.

Contrary to the proportional selection, this method is widely used for both practical and theoretical purposes.

4. Investigated selection methods

4.1. A histogram selection

This method is a result of searching for an algorithm of selection with a high ability to maintain the population diversity while preserving the selective pressure of population growth, as presented in [17]. The version presented below is a slightly modified variant and it is also used as an element of new methods described later in this work.

The first step of this method's operation involves distributing the population based occurring values of the fitness function, hence its similarity to create a histogram and its name. There are two possible variants of performing this step. One ignores solutions with values of fitness function the same as just included in the list. However, quite often it happens that in the population there are also solutions with the same values of the fitness function, but representing completely different solutions.

Ignoring this fact constitutes a significant simplification of the problem and may result in a considerable loss of population diversity. A good solution to this problem requires a comparison of encoded representations of individuals with the same values of fitness functions, and such a mechanism is applied in the considered histogram selection. A proper way to detect such situations remarkably depends on the method of encoding solutions and in many cases can be quite complicated if the encoding method used is ambiguous (several different individuals may encode the same solution). For instance, to solve an instance of the TSP (Traveling Salesman Problem) using EA where a list of visited cities is treated as a solution encoding, several lists may seem not similar at first glance but may encode the same solution, differing the starting point or the travel direction.

Thus, it is necessary to adopt a metric in the domain of coded solutions and accept certain criteria to distinguish solutions. Generally, in the case of binary or integer encoding it can be assumed that the solutions are different if they differ in at least one position. It is possible, however, to assume identical solutions which differ to a greater extent. In the case of real number encoding, it is necessary to apply a certain minimum distance between the solutions below which they are considered to be identical. It would be a factor similar to the crowding factor, widely used in EA.

Regardless of the method used (whether comparing solutions or not), the obtained list of selected individuals is usually shorter than the list of individuals in the parent population ($s \leq \mu + \lambda$ or $s \leq \lambda$), depending on the population development strategy due to possible repetitions of the same solutions in the population.

The second step of histogram selection can also be executed in two ways. The first considered version can be used as an independent selection method and is characterized by a relatively low level of selection pressure. Each individual from the list passes to the offspring population the number of individuals that results from rounded to the nearest integer ratio of its value of fitness function to the average of the fitness function for the list, multiplied by population size. This is illustrated by Formula (12). In this formula, a scaling mechanism is used to minimize rounding errors. If, despite this, the size of the descendant population is greater or less than the expected number of individuals, the population is replenished using the best individuals who have not yet entered the new generation, or the proper number of the weakest among the selected individuals is removed.

$$n_l(t+1) = f\left(s \cdot \frac{F_l(t)}{\sum_{j=1}^{s} F_j(t)} \cdot \frac{\mu}{\sum_{i=1}^{s} f(\frac{F_i(t) \cdot s}{\sum_{k=1}^{s} F_k(t)})}\right) \tag{12}$$

where:

$n_l(t+1)$ – the number of offspring, copies of the l-th solution from the list in the descendant population, where $l \in 1...s$,

$f(...)$ – a function that implements one method of converting the real value result to an integer value, it can be either round – which approximates the actual value to the nearest integer or floor – rejecting the fractional part of the calculated number of descendants,

s – the number of different values of the fitness function (or better, different genotypes) distinguished in the resulting list

$F_l(t)$ – the value of the fitness function for the l-th individual in the current population.

In the second version of histogram selection (flat histogram selection), the survival of the fittest individuals from the prepared list of individuals is performed in this step of the considered method. Thus, the descendant population is more diverse than in the first version of the second step, since it does not contain repeated solutions and even very good individuals are mentioned only once. This selection version has very small selective pressure; it mainly strongly increases the population diversity and rather cannot be used as an independent selection method but only as an auxiliary method or some form of population preprocessing.

4.2. Mixed selection

The histogram selection gives good results in evolutionary computations, preventing the too rapid convergence of the population, but it is characterized by a rather small selection pressure on the population towards promoting the best individuals, much smaller than the deterministic roulette method. It has a long or infinite time of unification of the population. On the contrary, the deterministic roulette method remarkably promotes the best solutions, but this leads to a rapid loss of population diversity and premature convergence. The unification time of the population can be estimated in this method as follows:

$$\tau \leqslant \frac{\ln(\mu)}{\ln(1+\frac{\lambda}{\mu})} \tag{13}$$

where:

τ – time (number of generations or iterations) to fill the population with identical individuals, without affecting the evolutionary operators,

μ – parent population size,

λ – the size of the new generation.

A combination of advantages of both methods, compensating for their shortcomings, can be achieved using a mixed selection. This idea was previously presented in [17]. The mixed selection consists of two component methods with significantly different properties: histogram selection (flat histogram selection or simple histogram selection) which has the property of significantly increasing the diversity of the population, and the deterministic roulette, with a strong focus on promoting the best individuals and thus decreasing the diversity of the population. These methods are randomly selected and executed during the operation of the evolutionary algorithm.

The probability of selection and performance of each method shows the formula (14):

$$\begin{aligned} p_{his}(t+1) &= \begin{cases} p_{his}(t) \cdot (1-a) \text{ for } R(t) > 3 \cdot \sigma(F(t)) \\ p_{his}(t) \cdot (1-a) + 0.5 \cdot a \text{ for } R(t) \geqslant 0.5 \cdot \sigma(F(t)) \text{ and } R(t) \leqslant 3 \cdot \sigma(F(t) \\ p_{his}(t) \cdot (1-a) + a \text{ for } R(t) < 0.5 \cdot \sigma(F(t)) \end{cases} \\ R(t) &= max(F_{av}(t) - F_{min}(t), F_{max}(t) - F_{av}(t)) \\ p_{det} &= 1 - p_{his} \end{aligned} \tag{14}$$

where:

$p_{his}(t)$ – the probability of histogram selection,
$p_{det}(t)$ – the probability of selection by deterministic roulette,
$F_{av}(t)$, $F_{min}(t)$, $F_{max}(t)$ – average, minimum, and maximum values of the fitness function in the population,
$\delta(F(t))$ – the standard deviation of the population fitness function.

If the set of values of the fitness function for the population is characterized by too small standard deviation ($\delta(F(t))$) in relation to the span of the fitness function values ($max(F_{av}(t) - F_{min}(t), F_{max}(t) - F_{av}(t))$), then the desired operation is to increase in the probability of histogram selection (third position in the formula (14)). Otherwise, it desired to increase the probability of selection by a deterministic roulette method (first position in formula (14)). If the parameters of the population are within the range considered preferable, the probabilities of selection of both methods are nearly equal (the second entry in the formula (14)). It should be noticed that the rule must always be fulfilled: $p_{his}(t) + p_{det}(t) = 1$, i.e., any of the methods must always occur.

4.3. Interval selection

This method is based on the division of the parental population into several subpopulations. The criterion for the division are values of the fitness function of individuals. At the beginning, a sorted list of individuals with different values of fitness function in a population is created. In many cases, equal values of the fitness function may characterize different solutions (individuals). This selection method deals with such a case and additional items are created on the list, including different individuals with the same values of fitness function. A whole range of values of this function occurring is divided into some number of compartments. This number of compartments is one of the parameters of the method. The most common distribution used in simulations contains three subpopulations, with the following parameters:

- individuals assessed at 90 % – 100 % of the best value of the fitness function;
- individuals assessed at 50 % – 90 % of the best value of the fitness function;
- individuals assessed at 10 % – 50 % of the best value of the fitness function.

Each compartment is guaranteed a certain percentage of individuals that will appear in the next parent population. Of course, the worse the interval of fitness function values, the fewer should be a guaranteed percentage of the next parent population participation.

Percentage distribution of the number of descendants for each compartment can also be adapted to the requirements of the problem being solved. In the conducted computer simulations, the best results were obtained using the following schedule:

- The first interval (the best individuals) receives 60 % of the descendant population.
- The second interval (average individuals) provides 30 % of individuals of the descendant population.
- The third interval (weakest individuals) receives 10 % of the descendant population.

The sum of all pools of the intervals forms the entire new population. The selection of individuals that pass to the next population within a compartment may be carried out in various ways. In the presented results of the computer simulations, the selection of the best individuals was used, but of course, several well-known methods for instance tournament selection or other conventional methods, can be used.

The applied distribution of the parent population (the number of used compartments and their percentage distribution) and allocated pools of seats to be filled in the next population are method parameters and can be modified, not only a priori but also during the algorithm run-time. With such modifications, it is possible to change the profile of the desired descendant population and significantly modify the properties of the method of selection of individuals.

It should be noted that this method is applicable in cases where the descendant portion of the population (λ) is greater than the parent part (μ) or selection is made from both parts ($\mu + \lambda$) to have a suitable subject pool from which to select.

4.4. Selection with lifetime and taboo list

The problem of stagnation in the evolutionary computation after reaching a certain level of solutions associated with the stagnation in local optima is widely known and very difficult to control in evolutionary algorithms. There are many ways to deal with this problem, from "killing" all parent individuals (even if are better than their offspring) in non-elitist methods, through methods that allow continuous monitoring of the diversity of the population and insert the newly drawn solutions to maintain a high diversity of the population to controlled selection methods. The presented in this point method is a kind of controlled selection, combining several previously used solutions in different areas of artificial intelligence in one – a selection using the lifetime of an individual and an array/list of taboo solutions. The lifetime of an individual was already proposed in slightly different versions – as an equivalent of the selection method in the EA with varying population sizes in the work Arabas [2]. The taboo list is used in several versions of methods for solving optimization problems called taboo search, where it is used to store a variety of information about the discarded for some time solutions, their components, or methods of modification of solutions. This new method of selection can best be characterized as follows: the method of selection of individuals to the next parent population can be any previously known method

of selection, while the proposed method is responsible for the initial preprocessing of the population.

The creation of a new solution as a result of the random generation of the initial population, or as an effect of reproduction using genetic operators, is associated with giving it a certain value, which is a maximum lifetime in iterations (generations) of the evolutionary algorithm. This value may depend on the quality of an individual, but it is not required. It is not a guaranteed lifetime, the solution may eliminate earlier the selection method used. During its lifetime, the solution can contribute to the creation of new solutions, but maximally after its lifetime, it is eliminated from the population and can be inserted into the taboo list. It enrolls solutions with appropriate parameters: they must have good values of the fitness function (in the range of 60 % -100 % of the current best) and cannot be too similar to those already on the list (in terms of content code of compared solutions). The issue of assessing the similarity of solutions has already been mentioned while describing the histogram selection, here it looks the same, but the similarity of solutions placed on the taboo list has to be even more distant (in the Hamming sense, for instance). It contains rather certain classes or patterns of similar solutions, i.e. not solutions with differences in the one position, but bigger. It must be noticed that with the taboo list also rejected solutions have a certain effect on the form of a population of solutions because any solution that is too similar (in this case the differences of similarity threshold may be smaller than when placed on the list) to the existing on the taboo list is eliminated from the population, even if it is still "young". The taboo list is reset after finding the next best solution. The taboo list becomes active during the stagnation of calculations. If evolutionary computations frequently lead to the discovery of new and better solutions, the list remains inactive. This allows to "beat in" the population from a local optimum if necessary, requiring no additional effect and delay of the calculation, when it is not needed.

5. Properties of proposed methods in computer simulations

Theoretical formulas for coefficients measuring selection characteristics are rather difficult to obtain for more complicated methods, although, there are some results for standard ones [6, 10, 15, 16, 22]. Thus, a set of practical simulations can show results obtained for the selection methods presented in this study, based on solved computational problems. This method of properties investigation is not such universal as derived formulas, but despite all, can tell a lot about the properties of proposed selection methods.

5.1. Sample computational problems for testing described selection methods

As a testing base, an evolutionary algorithm solving the maximum α-clique searching problem has been chosen. An α-clique is a generalization of a clique notion. The clique

in a graph is every complete subgraph of the whole graph. The complete subgraph is a subgraph where all vertices have edges between them. For sparse graphs, cliques are rather small and are too highly connected structures to find locally proper connected centers, for instance, real transportation networks. Some kind of more flexible and controllable structure would be better to divide the whole graph and find locally interconnected structures. This elementary brick for graph partitioning could be an α-clique. The α-clique is defined in [18] or [14]:

Let $A = (V', E')$ be a subgraph of graph $G = (V, E)$, $V' \subseteq V$, $E' \subseteq E$, $k = Card(V')$ and let k_i be a number of vertices $v_j \in V'$ that v_i, $v_j \in E'$.

1. For $k = 1$ the subgraph A of graph G is an α-clique with desired value of α.
2. For $k > 1$ the subgraph A of graph G is an α-clique with the desired value of α if for all vertices $v \in V'$ fulfill the condition $\alpha \leqslant \frac{k_i+1}{k}$, where $\alpha \in (0, 1]$.

The α-clique is simply a subgraph, where all nodes are connected with not less than $\alpha \cdot 100\%$ of all their nodes (the subgraph size), with α representing the desired connectivity percentage. Of course, the α-clique with $\alpha = 1$ is simply a clique.

From fundamental basic graph properties, it can be deduced that for $\alpha > 0.5$ obtained α-cliques must constitute connected subgraphs. A connected (sub)graph is a kind of graph, where for each pair of vertices there is a path (a continuous sequence of edges) between them and this is the most interesting case because it is not good to derive transportation centers (often called hubs) with isolated, not connected vertices.

Similarly to the clique case, also it can be useful to find the maximum α-clique in a graph. It is a non-trivial task, practically harder than finding the maximum clique problem (the problem of finding the maximum clique is NPH [1]), because not every subgraph of α-clique with imposed α is an α-clique with the same value of α, very often the value of α in a sub-α-clique is lower than in the whole α-clique. In the case of the clique, all subgraphs of this clique are also cliques. Thus, it is difficult to prepare a simple algorithm trying to find the biggest α-clique by adding nodes to the obtained one, because one can never know how many and which of them must be added. In the case of a clique, one doesn't know only which one may be added, if it is possible to make it bigger, but if it is possible, bigger cliques can be obtained by adding one new vertex in one step. In the case of α-clique, this is often not possible. This fact means that the majority of efficient approximate algorithms prepared to find the maximum cliques cannot be extended to search for bigger or maximum α-cliques. Thus, it seems justified to use the evolutionary algorithm to solve the problem, because this method can do it efficiently.

As the second computational test problem, the widely known classical TSP was considered. It was used only in several cases to show some interesting properties of selection methods that led to the observation that properties of selection methods also depend on the solved problem.

5.2. Evolutionary algorithm used to solve the problem

5.2.1. The maximum α-clique problem

The information about the considered graph is stored in a square neighborhood matrix that describes connections of all graph nodes: 0 – no connection, 1 – presence of connection. Because a single node is also treated as an α-clique, the matrix has 1 on the main diagonal. The value of α (the α-clique parameter) is imposed as the problem parameter.

Each member of the EA population encodes the solution to the problem as a variable-length vector representing the biggest α-clique derived by that member. The not selected nodes form a similar vector, serving as a repository of potential new nodes that can be attached to the derived α-clique. Additionally, each member of the population contains more data. This includes a vector of real numbers, which describes its knowledge about genetic operators and the operator number chosen to modify the solution in the current iteration. More details about genetic operators and the method of evaluation, selection, and application of them will be given later in this chapter.

The described data structure requires specialized genetic operators, which modify the population of solutions. Each operator is designed in such a manner that it preserves the property of being an α-clique with the desired value of α for the modified solutions (after its application, the actual value of α for the modified solution is computed). If a modified solution violates the limitation of being an α-clique, the operation is canceled and no modification of the solution is performed. While this method makes it more challenging for the evolutionary algorithm to find satisfactory solutions, potentially encountering greater difficulties with local extrema compared to, for instance, a method with a penalty function, it ensures that the computed solutions are admissible.

Designed genetic operators are:

- mutation – an exchange of randomly chosen nodes in α-clique and the storage vector,
- transfer of randomly chosen node from the storage to the α-clique,
- "intelligent" movement – performed only if this modification gives a better value of fitness function,
- concatenation – this operator tries to concatenate small vectors chosen from the storage with α-clique,
- multiple versions of operators are also applied (randomly selected numbers of repetitions of the genetic operator in one generation).

5.2.2. The TSP problem

Solutions in this case are encoded as lists of cities to be visited in the order described by the list. Several genetic operators were used to solve the problem: random and

heuristic ones. All operators ensure that the generated offspring can consistently encode feasible solutions. The operators are:

- mutation – a random exchange of two cities in the list of cities,
- crossover – starting from the first city of one list, the next cities are chosen in turn from one or second list (the city closer to the previously accepted city is selected from the parent individual and introduced to the offspring),
- inversion – a randomly chosen fragment of the list is used in the reverse order,
- transposition – a randomly chosen fragment of the list of cities is moved to another place (also randomly chosen) 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 k-optimal method [21] for approximate solving of TSP),
- neighborhood-1 operator – exchanges a randomly chosen city in the route for another, randomly chosen from the list of the closest ones in the geometric sense (a list of several closest cities is generated for every city during the initialization of the algorithm),
- neighborhood-2 operator – takes a city close to the selected one from the path, moves all cities between them by one position, and inserts the picked city next to the selected one.

5.2.3. The common part of used evolutionary algorithms

The application of several specialized genetic operators requires a method that selects and executes them during evolutionary computations. In the approach used in [17], it is assumed that only one, selected operator modifies the solution in one generation (not two as in typical EA). In that case, it is easy to allocate the result of that operation to the individual and operator. Thus, the operator obtaining better results should have a higher probability and more frequently affect the population than the worse one. But it is very likely that the operator, that is proper for one individual, gives worse effects for different solutions because of its location in the search space.

Thus, each individual may have its preferences, enabling it to select operators that align with its specific characteristics or requirements. For example, an individual might favor operators that excel in certain regions of the search space or demonstrate effectiveness for specific types of solutions. To obtain this possibility, every individual has a vector of quality factors, where each factor corresponds to one genetic operation and is a measure of the quality of that operator. The normalized vector of qualities becomes a base to compute the probabilities of selection and execution of genetic operators by population members (15).

This set of probabilities – a base of population experience can be inherited and improved over the next generations.

$$p_{ij}(t) = \frac{q_{ij}(t)}{\sum_{i=1}^{L(t)} q_{ij}(t)} \tag{15}$$

where:

p_{ij} – represents the probability of execution of the genetic operator,
$q_{ij}(t)$ – represents a quality factor of the genetic operator,
$L(t)$ – represents the actual number of genetic operators (in some evolutionary algorithms may vary during computations),
t – represents the current time.

The method to compute the quality factors is based on reinforcement learning [8, 20] (one of the algorithms used in machine learning). An individual is treated as an agent whose role is to select and call one of the evolutionary operators. When the selected i-th operator is applied, it can be regarded as an agent's action a_i leading to a new state s_i which in this case is a new, modified solution. The agent receives a reward or a penalty respective to the quality of the new state (solution). The aim of the agent is to perform the actions that give the highest long-term discounted cumulative reward V^*. The described activity leads to the Q-learning technique of temporal reward assignments, which can be written as:

$$V(s_{t+1}) = V(s_t) + \beta(r_t + \gamma V^*(s_{t+1}) - V(s_t)) \tag{16}$$

where:

$V(s_t)$ – is a quality factor or discounted cumulative reward, that can be identified with q_{ij} from (15),
$V^*(s_{t+1})$ – estimated value of the best quality factor (in the experiments the value gained by the best operator was taken),
β – is a learning factor,
γ – is a discount factor,
r_t – represents the reward for the best action, which is equal to the improvement of the quality of a solution after execution of the evolutionary operator,
t – index of the current moment in time.

Selection methods used in the presented evolutionary algorithm to obtain shown further results were described in sections 3 and 4.

5.3. The testing problem data

Unfortunately, it is not possible to obtain the testing data (data with known optimal solution) for the maximum α-clique problem from any data repositories. Instead, it is possible to find several problems for the maximum clique. Since the clique is a special case of α-clique with $\alpha = 1$, the problems for the maximum clique found in BHOSLIB: Benchmarks with Hidden Optimum Solutions for Graph Problems (Maximum Clique, Maximum Independent Set, Minimum Vertex Cover and Vertex Coloring) – Hiding Exact Solutions in Random Graphs [4] have been used. The chosen problem was

a graph with 4000 vertices and 7 425 226 edges with the maximum clique equal to 100 (frb100-40.clq.gz). In all cases, starting populations are generated using a simple greedy algorithm, which randomly chooses the first node and tries to add to the obtained α-clique remaining graph nodes in random sequence. This method is, as it was said, rather a poor optimization method but is quite fast and generates different solutions with a size of about 70% of the size of known the maximum clique, which can be a good starting point for the evolutionary search of better ones.

The second considered problem is a classic TSP problem with 1002 cities (pr1002.tsp) [9] with the optimal solution 259045. In this case, also a greedy algorithm was used to generate the initial population of solutions.

5.4. Results obtained for selected coefficients of selection characteristics

5.4.1. The takeover time

The takeover time shows the strength of the selective pressure of a selection method, showing how fast the whole population would converge to one, the best solution, assuming the lack of genetic operators and preserving one copy of the best individual. For simpler methods (proportional/roulette selection, deterministic roulette, tournament selection) this time can be described analytically, but for more complicated ones this is difficult or maybe not possible. Thus, in this work, values obtained from real computer simulations are presented. Table 1 presents averaged results obtained from 10 simulations. Simulations lasted maximally 100 epochs, if during this time the population hadn't converged, the takeover time was assumed to an infinite.

Table 1
Obtained takeover times for investigated selection methods

Selection method	Roulette / Roulette with elitism[3]	Deterministic roulette	Tournament	Histogram	Mixed: det.+hist.
Takeover time (epochs)	15.6	7.8	9.4	∞	6.8
Selection method	Histogram flat	Mixed: det.+hist. flat	Interval	Mixed: det.+hist. flat with lifetime and taboo	–
Takeover time (epochs)	∞	6.2	∞	8.8	–

[3]In this case it is the same selection method, the investigation of takeover time assumes preserving the best individual in the population (elitism).

As it can be seen, selections with the heuristic population control mechanism (histogram, histogram flat, and interval) have infinite takeover time, traditional ones have takeover times from 6.8 (roulette) to 9.4 (tournament), mixed selections have takeover time similar to the component with stronger selection pressure, but this value depends on the probability of execution of both component methods, which can be seen in Table 2.

Table 2
Obtained takeover times for different probabilities of component selection methods in mixed selection

	Mixed: det.+hist.	Mixed: det.+hist. flat	Mixed: det.+hist. flat with lifetime and taboo
$p_{his} = 0.95$	∞	34.4	∞
$p_{his} = 0.85$	41.0	10.8	29.6
$p_{his} = 0.75$	29.4	11.6	15.8
$p_{his} = 0.65$	19.0	7.8	11.4
$p_{his} = 0.55$	14.4	7.8	9.0
$p_{his} = 0.45$	6.8	6.2	8.8

For the selection with lifetime and taboo list, the value of this parameter is not a proper measure of selection properties, because for established longer lifetime than the takeover time (this situation is presented in Tables 1 and 2, where the lifetime is set to 100), there is no influence of it on the obtained takeover value. For shorter than takeover lifetimes imposed, all the population members should be killed (no new individuals are created since the genetic operators are disabled) after this time and only the best individual is artificially preserved to fulfill the conditions of the coefficient measure, thus the takeover time is equal to the imposed maximal lifetime of individuals. This is not the typical situation that this selection method works and obtained in this case value would say nothing about this selection properties. This selection method is prepared for long-time computations and in this case, its properties can be observed. Thus, the simple conclusion is that the takeover time is not a good measure of properties for more sophisticated selection methods.

5.4.2. The loss of diversity caused by the selection process

The *loss of diversity* is a harmful phenomenon, especially in small populations (such as typically used in evolutionary computations), because it can lead to the premature convergence of the population to local minima. If the mutation level is rather small, a crossover of almost identical individuals produces almost the same individuals and the progress in computations is negligible. Even if more sophisticated operators are used, similar phenomena occur. But of course, selection is necessary to continue the process of evolution and better individuals should have more descendants than worse.

As in other aspects of evolutionary algorithms, it is important to keep the balance between the diversity of the population removing the worst and replication the best solutions. Unfortunately, the formula for the optimal composition of the population is not known. We can only make experiments with different types of selection and evaluate them, comparing obtained results for the solved problem. But this would allow only to see which one is better for the given instance of the solved problem.

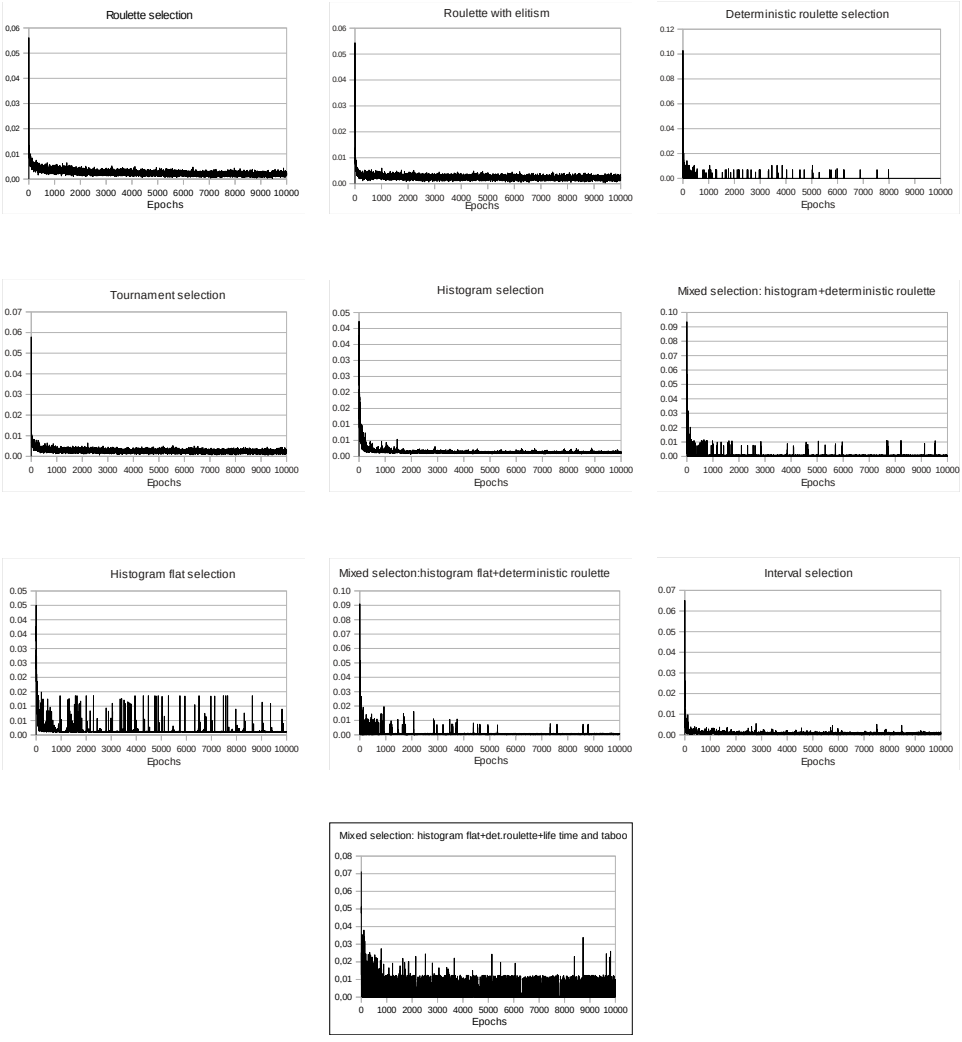

Figure 1. Graphs of average values of the loss of diversity factor for several tested selection methods

Table 3
The comparison of the average value of the loss of diversity factor in the maximum α-clique problem using tested selection methods

Selection method				
Roulette	Roulette with elitism	Deterministic roulette	Tournament	Histogram
0.00252	0.00244	0.00017	0.00259	0.00148
Mixed: det.+hist.	Histogram flat	Mixed: det.+hist. flat	Interval	Mixed: det.+hist. flat with lifetime and taboo
0.00075	0.00139	0.00070	0.00076	0.00128

The values collected and presented in Table 3 are the average values of 10 simulations (10,000 epochs) recorded at each epoch using formula (4). The common part of populations considers not only the fitness function value but also the encoded solutions. Individuals with identical values of fitness function but encoding different solutions are treated as different.

As it can be noticed, the smallest values of the *loss of diversity* factor (the most similar populations before and after selection) have: a mixed selection consisting of histogram flat and deterministic roulette with lifetime and taboo, interval selection, and roulette selection, but only the first one gives also very good results in problem-solving (see Tables 4 and 5). It suggests that small values of the *loss of diversity* factor are important, but it is not the most decisive factor that influences the evolution process.

5.4.3. The selection intensity

The selection intensity factor shows the convergence properties of the investigated methods. For simpler ones, there are some theoretical results [7, 13], for more sophisticated and heuristic selection methods only practical experiments can show their behavior. The results obtained during simulations are collected in Figure 2.

As can be noticed, several selection methods like roulette with elitism, deterministic roulette, tournament selection, histogram selection, histogram flat selection and mixed selections consisting of histogram ones with deterministic roulette have the property of converging their selection intensity at a value close to 0.2, but roulette selection, interval selection, and mixed selection consisting of histogram flat, deterministic roulette with lifetime and taboo do not converge. It means that there are strong changes in population composition during the whole algorithm simulation, while the converging ones have rather small changes.

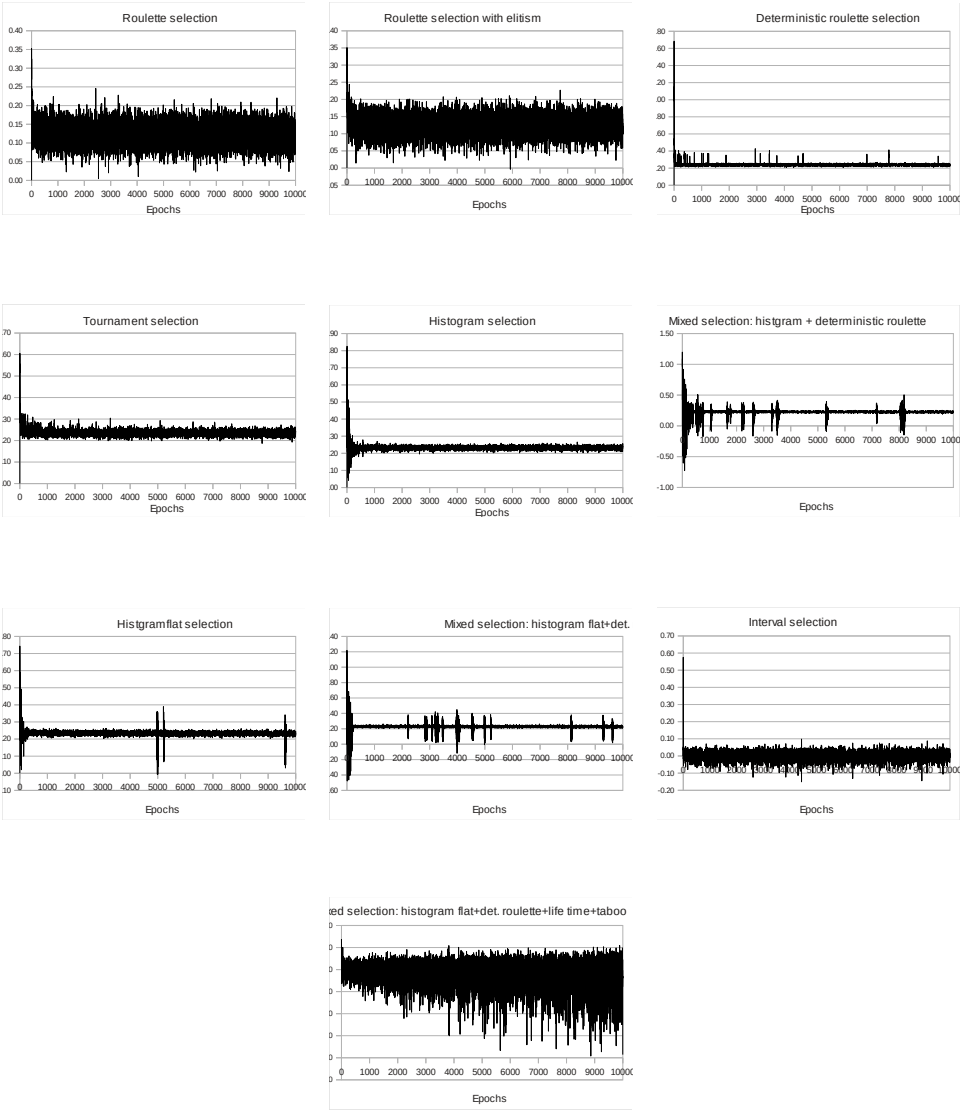

Figure 2. Graphs of average values of the selection intensity factor for several tested selection methods

5.4.4. The selection variance

The *selection variance* (3) also describes the properties of selection methods, in this case, it is possible to trace how changes the variance of evaluations of solutions in the population as a result of the selection process. The obtained results are presented in Figure 3.

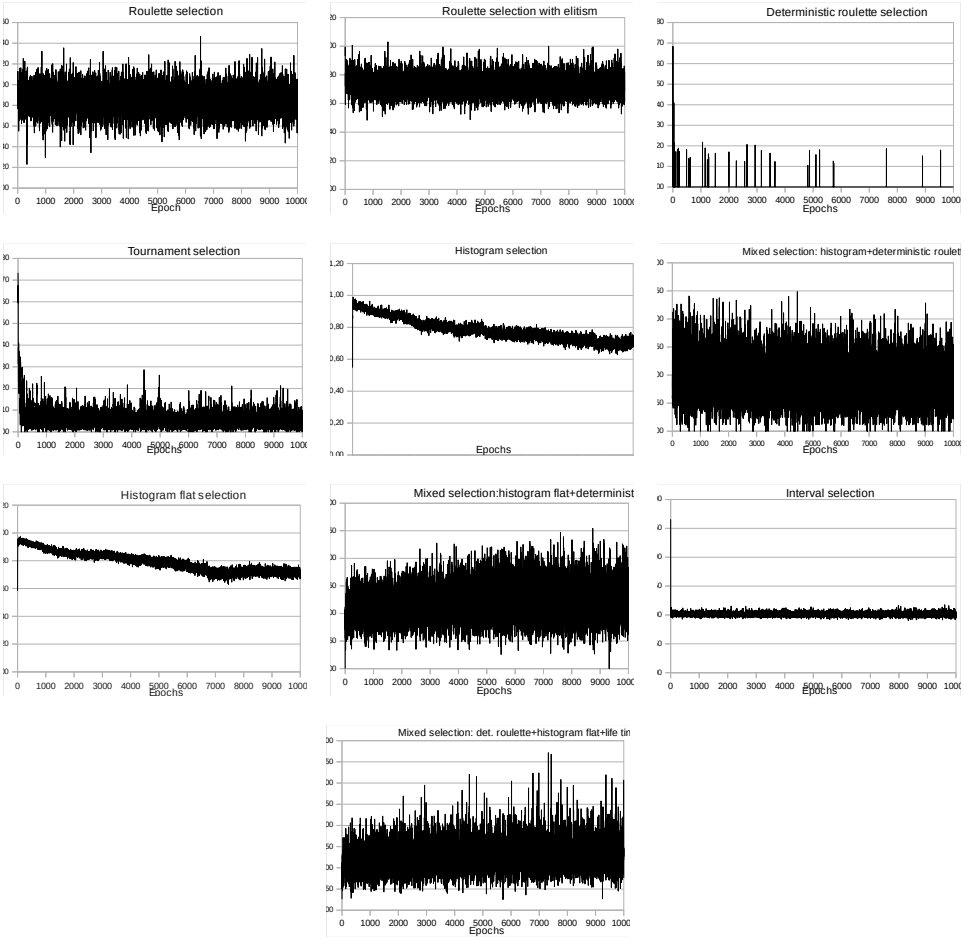

Figure 3. Graphs of average values of the selection variance for several tested selection methods

5.4.5. The population diversity before (s_b) and after (s_a) selection

The diversity of the population before and after selection can also be a measure of its properties. It is widely known that higher population diversity favors better results achieved by the evolutionary algorithm due to the prevention of premature convergence of the population of solutions. The presented further results were obtained for the maximum α-clique problem with $(\mu+\lambda)$ strategy[4] of the population development with $\mu = 100$ and $\lambda = 700$. The average values of population diversity as defined in (5) and (6) are presented in Table 4.

[4]It means that the new parent population was selected from the old parent and offspring populations.

Table 4
Values of average population diversity coefficients s_b and s_a obtained using investigated selection methods for the maximum α-clique problem

	Selection method				
	Roulette	Roulette with elitism	Deterministic roulette	Tournament	Histogram
s_b	0.0222	0.0200	0.1052	0.0215	0.1332
s_a	0.0946	0.0844	0.7834	0.0912	0.9998
	Mixed: det.+hist.	Hist. flat	Mixed: det.+hist. flat	Interval	Mixed: det.+hist. flat with lifetime and taboo
s_b	0.1302	0.1331	0.1286	0.1364	0.0661
s_a	0.9838	1.0000	0.9690	0.8948	0.4865

Table 5
Results of EA simulations obtained using investigated selection methods for the maximum α-clique problem (average sizes of obtained α-cliques)

Iteration	Selection method				
	Roulette	Roulette with elitism	Determinstc roulette	Tournament	Histogram
0	70.7	70.5	70.9	71.0	70.7
10	71.1	71.3	72.0	71.4	73.2
100	73.0	74.4	77.1	74.2	77.6
1000	78.4	78.5	81.8	79.6	81.1
10000	84.0	81.8	85.9	84.3	84.2
100000	87.5	82.7	87.6	88.0	85.6
Iteration	Mixed: det.+hist.	Histogram flat	Mixed: det.+hist. flat	Interval	Mixed: det.+hist. flat with lifetime and taboo
0	70.8	70.7	70.4	70.5	70.8
10	73.4	72.2	72.4	71.7	72.1
100	78.8	75.7	79.9	75.5	77.4
1000	84.7	79.0	85.3	78.4	84.3
10000	87.7	81.3	88.4	80.5	88.4
100000	88.3	82.8	88.9	83.6	91.0

Comparing these values with obtained solutions (Tab. 5) it can be seen that the highest values of population diversity (histogram flat, histogram) are not necessarily bounded with the best results obtained. The same is true for the lowest values of population diversity.

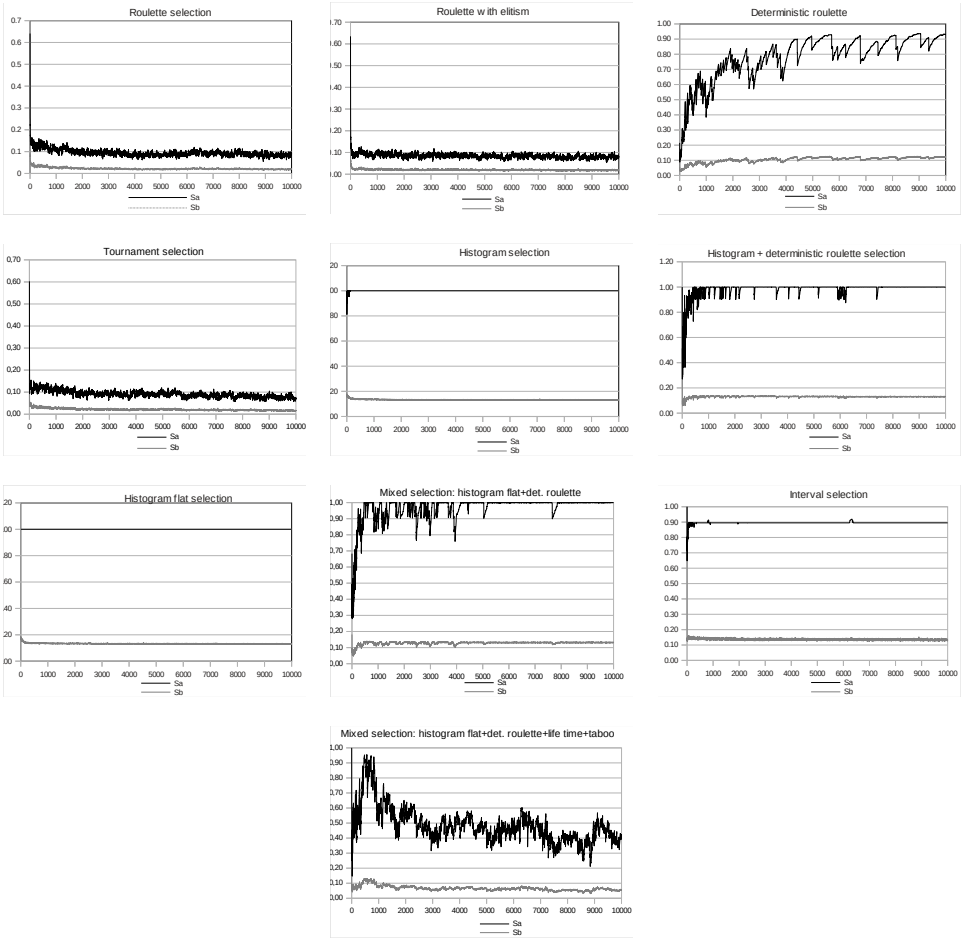

Figure 4. Graphs of average values of the diversity before (s_b) and after (s_a) selection for several tested selection methods

5.5. Values of obtained solutions for solved problems using investigated selection methods

Tables 5 and 6 show the averaged results of 10 computer simulations for several selected computation stages. As it can be seen, mixed selections outperform all other methods and the best of them is mixed selection with lifetime and taboo (Tab. 5). Differences among them are not very big, but noticeable. Similar conclusions can be drawn by analyzing Table 6, where the best results are obtained using mixed selection consisting of deterministic roulette and histogram flat selections, but the version with lifetime and taboo gives only slightly worse results.

The worst results are obtained using the roulette method (no improvement of obtained results has been recorded) and interval selection (Tab. 5), for the TSP problem (Tab. 6) the roulette method is significantly worse than other methods. This is not a surprise, because the roulette method is rather used for theoretical, not practical purposes. The interval selection method has been prepared for non-stationary optimization tasks and in this kind of optimization problem it works very well (see [19]) but it is not as good for stationary problems.

Table 6
Results of EA simulations obtained using investigated selection methods for the TSP problem (average distances)

Iteration	Selection method				
	Roulette	Roulette with elitism	Determinstc roulette	Tournament	Histogram
0	292616.4	293119.9	292608.4	292511.5	292038.7
10	292616.4	292926.4	291143.0	292511.5	291178.2
100	292616.4	290326.4	281564.4	292511.5	281264.4
1000	292616.4	281528.9	272017.7	292511.5	271265.7
10000	292616.4	274079.4	271535.8	275166.7	270780.7
100000	292616.4	272065.9	270918.2	268827.5	270320.8
Iteration	Mixed: det.+hist.	Histogram flat	Mixed: det.+hist. flat	Interval	Mixed: det.+hist. flat with lifetime and taboo
0	291958.8	292427.0	292124.7	292529.8	292003.8
10	290503.3	291543.8	291140.0	292125.2	290449.4
100	279740.3	283248.2	282115.1	285322.9	281310.9
1000	270646.6	271432.7	271620.0	273924.4	271636.1
10000	270085.0	270096.3	270033.5	270089.8	270287.9
100000	269598.5	269728.8	268197.1	268420.4	268291.6

The comparison of average computation times of computer simulations (Tab. 7) shows that new selection methods are rather fast and also the times of their operation are competitive with those used traditionally.

Table 7
The comparison of average computation times (in seconds) of evolutionary simulations duration using tested selection methods for solved problems

Solved problem	Selection method				
	Roulette	Roulette with elitism	Deterministic roulette	Tournament	Histogram
α-clique	234796	184709	219306	214365	211530
TSP	1160.4	1247.3	761.4	1114.5	1176.6
Solved problem	Mixed: det.+hist.	Histogram flat	Mixed: det.+hist. flat	Interval	Mixed: det.+hist. flat with lifetime and taboo
α-clique	173653	189012	186148	188338	83099
TSP	1048.7	1012.7	775.2	7080.7	1103.0

6. Conclusions

The selection in nature is the main (and maybe even the only) force that directs the population development. Effects of its slow and simple but powerful activity can be seen everywhere, admiring the variety of animal and plant species. Artificial selection methods, which are usually used in evolutionary algorithms, have far less time to achieve the desired results, although the results don't have to be so spectacular as in the case of the natural one, but they should be quick and effective. Unfortunately, selection methods, usually used in EA, are mainly quite simple because they try to mimic the natural one, without any possibilities of tuning, control, learning, and adaptation. Presented in this article methods try to overcome this problem and improve this important part of almost every EA to enable more predictable, resistant to local optima, flexible, and faster behavior of used selection methods.

References

[1] Aho A.V., Hopcroft J.E., Ullman J.D.: *The Design and Analysis of Computer Algorithm*, Addison-Wesley Publishing Company, 1974.

[2] Arabas J., Michalewicz Z., Mulawka J.: GAVaPS – a genetic algorithm with varying population size. In: *Proceedings of The First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, vol. 1, pp. 73–78, 1994.

[3] Back T.: Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms. In: *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, vol. 1, pp. 57–62, 1994.

[4] BHOSLIB – Network Data Repository. http://www.nlsde.buaa.edu.cn/kexu/benchmarks/graph-benchmarks.htm.

[5] Blickle T., Thiele L.: *A Comparison of Selection Schemes used in Genetic Algorithms*, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), 1995. TIK-Report Nr. 11.

[6] Blickle T., Thiele L.: A Mathematical Analysis of Tournament Selection. In: *Proceedings of the 6th International Conference on Genetic Algorithms*, vol. 1, pp. 9–16, 1995.

[7] Cantú-Paz E.: Selection Intensity in Genetic Algorithms with Generation Gaps. In: *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation (GECCO'00)*, vol. 1, pp. 911–918, 2000.

[8] Cichosz P.: *Systemy uczące się*, WNT, Warszawa, 2000.

[9] ELIB: MP-TESTDATA – The TSPLIB Symmetric Traveling Salesman Problem Instances. http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html.

[10] Goldberg D.E., Deb K.: A comparative analysis of selection schemes used in genetic algorithms, *Foundations of Genetic Algorithms*, vol. 1, pp. 69–93, 1991.

[11] Holland J.J.: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, USA, 1992.

[12] Moscato P.: *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms.* Technical Report C3P 826, Caltech Con-Current Computation Program 158-79, California Institute of Technology, Pasadena, 1989.

[13] Muhlenbein H., Schlierkamp-Voosen D.: Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization, *Evolutionary Computation*, vol. 1(1), pp. 25–49, 1993. doi: 10.1162/evco.1993.1.1.25.

[14] Potrzebowski H., Stańczak J., Sęp K.: Separable Decomposition of Graph Using α-*cliques*. In: *Computer Recognition Systems 2*, vol. 1, pp. 386–393, 2007.

[15] Rogers A., Prugel-Bennett A.: Genetic Drift in Genetic Algorithm Selection Schemes. In: *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 298–303, 1999.

[16] Rudolph G.: Takeover Times and Probabilities of Non-Generational Selection Rules. In: *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, pp. 903–910, 2000.

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

[18] Stańczak J., Potrzebowski H., Sęp K.: Evolutionary approach to obtain graph covering by densely connected subgraphs, *Control and Cybernetics*, vol. 40(3), pp. 849–875, 2011.

[19] Stańczak J., Trojanowski K.: Non-stationary optimization with multi-population evolutionary algorithm. In: J. Arabas (ed.), *Evolutionary Computation and Global Optimization*, pp. 251–260, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 2007.

[20] Sutton R.S., Barto A.G.: *Reinforcement Learning: An Introduction,* 1st ed., MIT Press, USA, 1998.

[21] Sysło M.M., Deo N., Kowalik J.: *Discrete Optimization Algorithms with Pascal Programs*, Dover Publications, USA, 2006.

[22] Thierens D., Goldberg D.: Convergence Models of Genetic Algorithm Selection Schemes. In: Y. Davidor, H. Schwefel, R. Männer (eds.), *Parallel Problem Solving from Nature – PPSN III,* Lecture Notes in Computer Science, vol. 866, pp. 119–129, Springer, Berlin, Heidelberg, 1994. doi: 10.1007/3-540-58484-6_256.

[23] Zhong J., Hu X., Zhang J., Gu M.: Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms. In: *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, vol. 2, pp. 1115–1121, 2005. doi: 10.1109/CIMCA.2005.1631619.

Affiliations

Jarosław Stańczak
Systems Research Institute PAS, Newelska 6, 01-447 Warsaw, stanczak@ibspan.waw.pl

Received: 31.03.2023
Revised: 18.12.2023
Accepted: 18.12.2023