

UDC 519.8:519.6:004.8

Doi: 10.31772/2712-8970-2025-26-3-318-333

Для цитирования: Малашин И. П., Тынченко В. С. Применение методов кластерного анализа для динамической коррекции области поиска в генетическом алгоритме // Сибирский аэрокосмический журнал. 2025. Т. 26, № 3. С. 318–333. Doi: 10.31772/2712-8970-2025-26-3-318–333.

For citation: Malashin I. P., Tynchenko V. S. [Application of cluster analysis methods for dynamic search space adjustment in genetic algorithms]. *Siberian Aerospace Journal*. 2025, Vol. 26, No. 3, P. 318–333. Doi: 10.31772/2712-8970-2025-26-3-318–333.

Применение методов кластерного анализа для динамической коррекции области поиска в генетическом алгоритме

И. П. Малашин^{1*}, В. С. Тынченко²

¹Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)

Российская Федерация, 105005, г. Москва, ул. 2-я Бауманская, 5, стр. 1

²Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева
Российская Федерация, 660037, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31

*E-mail: ivan.p.malashin@gmail.com

Аннотация. В статье рассматривается применение методов кластерного анализа для повышения эффективности генетических алгоритмов при решении задач глобальной оптимизации многомерных функций, актуальных в ракетно-космической отрасли. Предмет исследования – динамическая коррекция области поиска генетического алгоритма (ГА) на основе статистической фильтрации кластеров индивидов. Тема исследования включает разработку метода динамической коррекции областей интереса по переменным задачи путём разделения популяции на группы с помощью алгоритмов кластеризации (с заранее определенным количеством, а также с динамически определяемым количеством кластеров), вычисления для каждой группы показателей численности и средней функции пригодности, и отсеечения кластеров, не вносящих значимого вклада в эволюционный процесс. Целью исследования является повышение скорости сходимости алгоритма без снижения качества результатов при решении задач смешанной оптимизации за счет эффективной адаптации области поиска на каждом шаге алгоритма. Эксперименты на тестовых функциях показали повышение скорости сходимости в среднем на 25–30 % по сравнению со стандартным алгоритмом. Полученные результаты демонстрируют возможность интеграции разработанного подхода в программно-аппаратные комплексы для автоматизированного проектирования ракетно-космических систем, что позволит снизить время расчётов и повысить точность выбора оптимальных параметров. Также предложенный подход может быть использован для выбора эффективных значений гиперпараметров широко спектра моделей машинного обучения, в частности, архитектурного синтеза искусственных нейронных сетей, различных топологий, включая глубокие нейронные сети и специализированные архитектуры.

Ключевые слова: эволюционные вычисления, глобальная оптимизация, генетические алгоритмы, область поиска, кластерный анализ.

Application of cluster analysis methods for dynamic search space adjustment in genetic algorithms

I. P. Malashin^{1*}, V. S. Tynchenko²

¹Bauman Moscow State Technical University (National Research University)
5, building 1, 2-ya Baumanskaya St., Moscow, 105005, Russian Federation

²Reshetnev Siberian State University of Science and Technology
31, Krasnoyarskii rabochii prospekt, Krasnoyarsk, 660037, Russian Federation

*E-mail: ivan.p.malashin@gmail.com

Abstract. This study investigates the application of cluster analysis techniques to improve the efficiency of genetic algorithms (GAs) in solving multidimensional global optimization problems, particularly those relevant to the aerospace industry. The research focuses on dynamic search space adjustment in GAs through statistical filtering of individual clusters. The proposed methodology involves: (1) developing a dynamic correction approach for variable domains by partitioning the population into clusters using both fixed-number clustering algorithms (*k*-means, *k*-medians, agglomerative, and spectral clustering) and density-based methods (DBSCAN); (2) evaluating cluster quality metrics including population size and average fitness; and (3) eliminating clusters that contribute insignificantly to the evolutionary process. The primary objective is to enhance algorithm convergence speed by 25–30 % (as demonstrated in benchmark testing) while maintaining solution quality in mixed optimization problems through effective search space adaptation at each iteration. The three-stage method comprises: (1) current population clustering, (2) elimination of clusters with below-average population size and fitness, and (3) dynamic boundary adjustment for remaining individuals' domains. Experimental results demonstrate the method's potential for integration into aerospace design systems, significantly reducing computation time while improving parameter optimization accuracy. Furthermore, the approach shows promise for hyperparameter optimization in various machine learning models, particularly in neural network architecture synthesis - including deep neural networks and specialized topologies.

Keywords: evolutionary computation, global optimization, genetic algorithms, search space adaptation, cluster analysis.

Introduction

Genetic algorithms (GAs) have proven to be a versatile tool for solving complex global optimisation problems in various fields of science and technology. In the rocket and space industry they are used for designing control systems [1], optimising the parameters of propulsion systems, aerodynamic shapes of wings and fairings, as well as flight trajectories and life support systems for spacecraft [2] and in many other tasks. The classical implementation of GAs supposes fixed search areas for each variable, which in the early stages can lead to excessive exploration of unpromising areas of *n*-dimensional space and slow the convergence process.

One of the priorities of the latest modifications has become reducing the amount of computation by allocating resources more economically and focusing on truly significant areas of the solution space. In order to achieve this, various approaches are used: the adaptation of parameters during evolution, training the selection of operator variants based on data from previous generations and dynamic search localisation that takes into account the current distribution of solutions. All of those things can allow accelerating convergence and increasing search efficiency without a significant increase in computational load.

This paper proposes a method for dynamic correction of the search area of a genetic algorithm, which differs from known methods in its implementation of an iterative population clustering procedure followed by the removal of unpromising clusters, which allows adapting effectively the search area and increasing convergence speed to the optimal solution. The essence of the method is that after

each selection step, the population is divided into groups using one of the clustering algorithms. The article studies the application of five approaches: with a predetermined number of clusters – k-means, k-median, agglomerative and spectral clustering, as well as the DBScan approach with a dynamically determined number of clusters. For each cluster, the number of individuals and their average suitability are calculated. The clusters that simultaneously did not reach threshold values for the search space correction and stopping criteria are completely excluded from further search; and the variable ranges are adjusted taking into account the distribution of the remaining individuals.

The main aim of the study is increasing the convergence rate without the decline in the quality of the found solution, which is important for complex engineering calculations in the field of rocket and space system design. The article presents a mathematical formulation of the problem, description of the clustering and filtering procedure, as well as the results of computational experiments on standard test functions that have 25–30 % saving in computational resources compared to a classical GA.

Review of existing methods

In recent years, there have been many modifications for GAs aimed at increasing their flexibility and eliminating premature convergence to local extremes. One of the key trends is implementing an automatic parameter adaptation. Instead of hard-coded mutation and crossing probabilities, modern algorithms evaluate population “diversity” (e.g., through genetic entropy or dispersion) and dynamically change these probabilities. In a number of implementations, the mutation step size itself evolves along with the solution (self-adaptive approach), which eliminates the need for lengthy setup during the preparation phase and helps to avoid premature compression of the population.

Therewith, multi-population or ‘island’ GA models have become increasingly widespread. In a traditional GA, there is a single population, but in modern implementations researchers have begun to create several small subpopulations (‘islands’) at once, each of which evolves relatively autonomously [4]. Individuals periodically migrate between islands: the best individuals flow into other populations. This allows combining the advantages of the parallel exploration of different regions of the solution space and the exchange of high-quality genes [5].

The integration with local optimisation methods has a significant role in the development of GAs, having generated memetic algorithms. The idea is the following: a genetic operator of global search is supplemented by powerful procedural local ‘further research’, for example, gradient descents (where possible) or direct local search algorithms in the absence of information about derivatives. As a result, a GA performs a broad ‘scanning’ of the space; and local algorithms improve the suitability value of prospective individuals [6].

Aside from that, multi-criteria optimisation has become a significant trend for the development of GAs, since in real-world engineering and economic problems it is necessary to take into consideration several competing aims simultaneously. Although the classical methods NSGA-II [7], MOEA/D [8] and SPEA2 [9] have become well established, in recent years specialists have proposed new strategies for better distribution of Pareto front solutions. For example, the adaptive distribution of reference points makes it possible to represent different front areas more evenly; and a dynamic crowding distance takes into account the local density of solutions: if the solutions are clustered too closely together, the algorithm automatically increases the separation distance, thereby encouraging exploration the areas that are less studied [10].

Surrogate modelling is actively developing, where objective function evaluations are replaced by predictive models (Gaussian Process, random forest, neural networks). In hybrid Surrogate-Assisted GA (SA-GA) schemes, data on the objective function is collected gradually: first, a simple surrogate is constructed, and then, as data accumulates, the algorithm retrains the model [11].

Finally, it is necessary to mention the integration of GAs with neuroevolutionary methods, where an algorithm not only optimises the weights of the neural network, but also its topology. The originally proposed NEAT (NeuroEvolution of Augmenting Topologies [12]) has acquired a new lease of life in combination with modern architectures (convolutional networks, transformers) [13]. In Deep Neuro-

evolution GA, a mutation operator can add or remove layers of the neural network, and crossover 'crosses' architectures, seeking to achieve a compromise between accuracy and compactness. Such approaches are being actively used in AutoML pipelines, where automatic neural network design reduces the time and effort required by engineers for manual setting.

Nevertheless, sufficient attention is not paid to the issues of search area correction [14], which confirms the relevance of the development of modern schemes for increasing the convergence speed of GAs through concentrating computational resources in promising regions.

Mathematical statement of the problem

The problem of the global optimisation of the multidimensional function of real variables on a limited domain is being considered. Let $\bar{X}$ be a vector of variables characterising the search space (1):

$$\bar{X} = \{x_i, i = 1, \dots, n\}, \quad (1)$$

where n is the dimensionality of the problem (the number of variables in the objective function), x_i is the i^{th} variable. For each variable an initial domain is specified (2):

$$x_i = D_{x_i} = [x_i^l, x_i^h], \quad (2)$$

where x_i^l and x_i^h are lower and upper bounds of the range of the acceptable values for the x_i variable. These bounds define the initial hyperparallelepiped search area in the n -dimensional space.

In the classical versions of genetic algorithms [15; 16], it is assumed that the bounds of D_{x_i} remain constant throughout the entire evolution process (3):

$$D_{x_i} = \text{const}, \forall i = 1, \dots, n. \quad (3)$$

Nevertheless, this assumption may be ineffective, since it does not allow taking into account the changing structure of decision distribution in the search process. This paper proposes and studies the method for dynamically correcting the search area of a genetic algorithm at each generation.

To formalise the proposed approach, let us introduce the following designation:

G is the total number of generations (iterations) of the genetic algorithm.

g is a current generation number, $g = \{1, 2, \dots, G\}$.

M is the number of individuals (decisions) in the population in each generation.

m is the index of a specific individual in the population, $m = \{1, 2, \dots, M\}$.

K is the number of clusters obtained after clustering the population.

k is a cluster index, $k = \{1, 2, \dots, K\}$.

Ind_m^g is the vector of parameters for the m^{th} individual in the g generation.

$F_{Ind_m^g}$ is the value of the fitness function for the Ind_m^g individual..

$IndSet_k^g$ is the set of all individuals belonging to the k cluster in the g generation.

$M_k^g = (IndSet_k^g)$ is the number of individuals in the k cluster in the g generation.

Each k cluster in the g generation is matched its own domain for each x_i variable (Fig. 1), which will be denoted as (4):

$$D_{x_i}^{g,k} = [x_i^{g,k;l}, x_i^{g,k;h}], \quad (4)$$

where $x_i^{g,k;l}$ and $x_i^{g,k;h}$ are current lower and upper bounds of the x_i variable in the k cluster in the g generation.

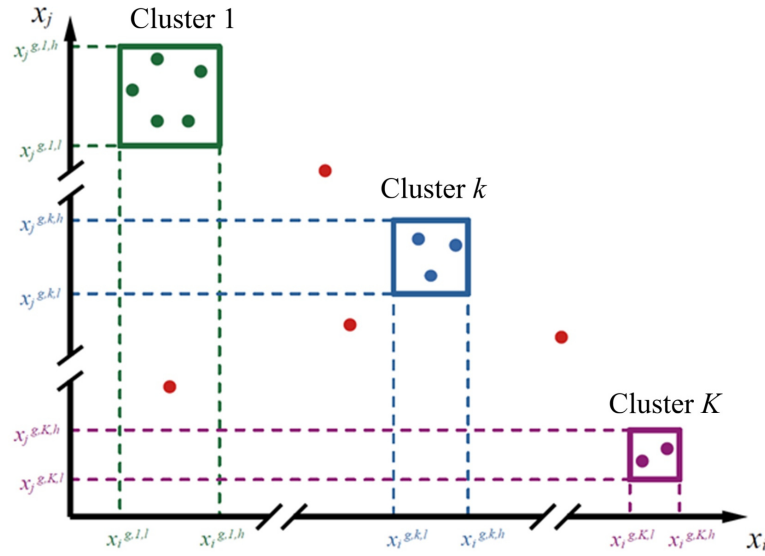


Рис. 1. Схема кластеризации

Fig. 1. Clustering diagram

The complete search area for the x_i variable on the g generation is defined as the union of all clusters (5):

$$D_{x_i}^g = \bigcup_{k=1}^K D_{x_i}^{g,k}. \quad (5)$$

If a cluster is considered to be ineffective (according to the criteria described below), the corresponding $D_{x_i}^{g,k}$ area is excluded from consideration, i.e. it can be assumed that (6):

$$D_{x_i}^{g,k} = \emptyset. \quad (6)$$

Clustering solutions

In order to increase the efficiency of searching in the solution space, cluster analysis is used. It allows identifying the promising areas of high-quality solutions for concentrating computational resources. This work uses both methods with a predefined number of clusters (K-Means, K-Medoids, agglomerative and spectral clustering) and a density algorithm that automatically determines the number of groups (DBScan).

Methods with a fixed number of clusters are based on minimising intra-cluster dispersion (K-Means), selecting medoids for robustness to outliers (K-Medoids), constructing a hierarchical dendrogram (agglomerative clustering), or using the spectral properties of a similarity matrix (spectral clustering). Their main limitation is the necessity to specify the number of groups in advance, which in practical tasks is selected using the ‘elbow method’ [17] or cross-validation based on the quality of solutions.

Density algorithms do not require the number of clusters to be specified in advance. DBScan identifies clusters as areas with a minimum number of neighbours within a distance that is not higher than ε (a parameter specified by a user), automatically discarding sparse points as noise.

Before clustering, it is necessary to normalise the data (bring each feature to a single scale, for example [0.1], and select a metric (Euclidean or Manhattan), since the difference in variable ranges and the method of calculating distances significantly influences the structure of clusters.

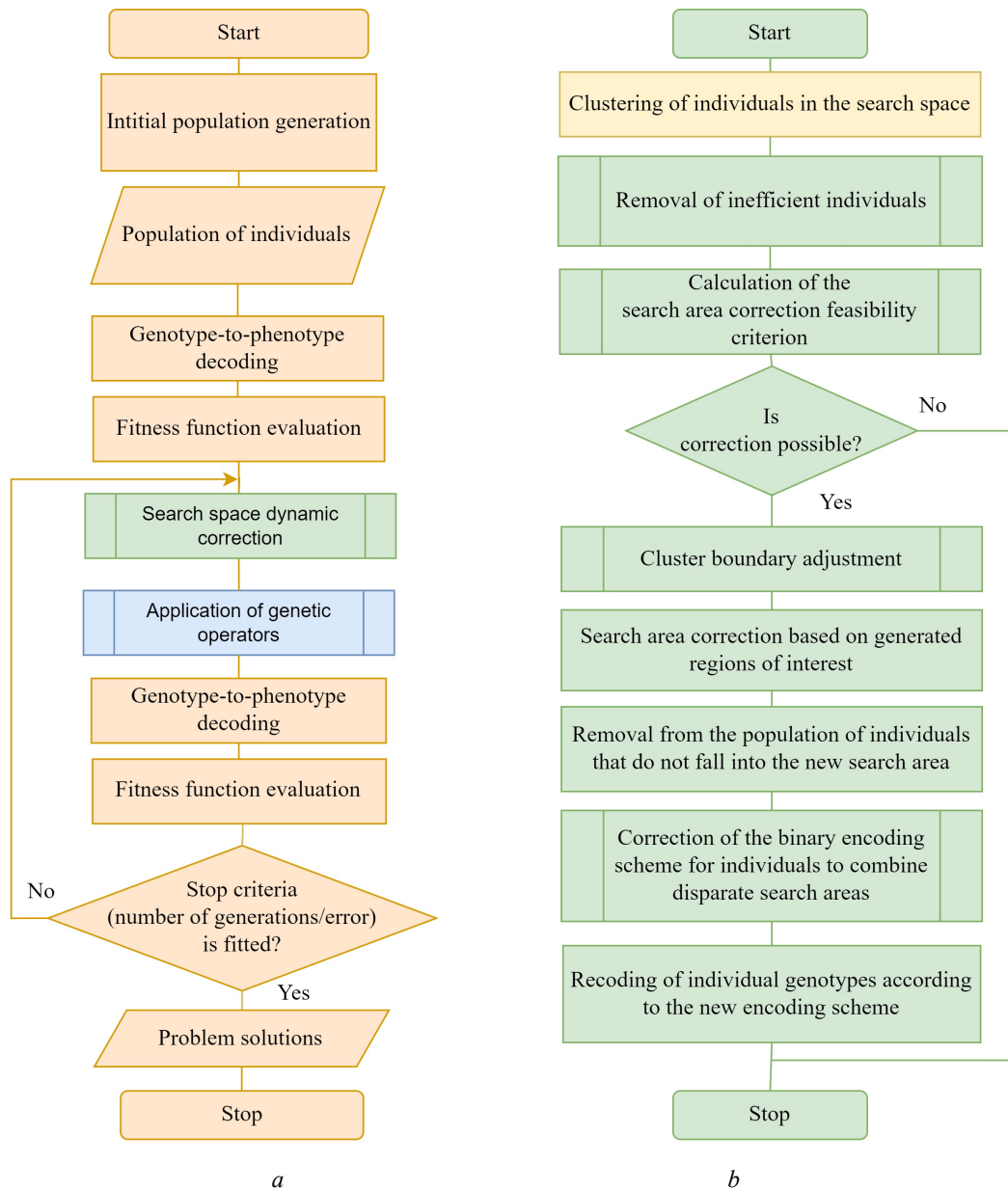


Рис. 2. Блок-схема:
a – ГА с динамической коррекцией области поиска;
b – процедуры динамической коррекции

Fig. 2. Block diagram:
a – GA with dynamic correction of search area;
b – block diagram of the dynamic correction procedure

In the optimisation algorithm (Fig. 2), the results of cluster partitioning are used for local correction of search area boundaries. In each cluster, corrective offsets are calculated based on the distribution statistics of individuals, which allows narrowing or expanding the search area around promising solutions. Furthermore, the probability of the recombination of the nearest representatives increases within clusters, which helps preserve the local features of the found optima; and computational resources are dynamically redistributed in favour of the clusters with higher solution quality or convergence speed. This approach provides a balanced combination of exploring and using solution space and contributes to faster and more reliable convergence of the algorithm.

Correcting cluster boundaries

After the clusters are formed, the boundaries of the search area in each cluster are corrected based on the analysis of the distribution of individuals within a cluster. Formally, the boundaries of the $D_{x_i}^{g,k}$ area are updated as follows (7):

$$D_{x_i}^{g,k} = \left[x_i^{g,k;l} + \Delta_{x_i^{g,k;l}}, x_i^{g,k;h} + \Delta_{x_i^{g,k;h}} \right], \quad (7)$$

where $\Delta_{x_i^{g,k;l}} \in R$ and $\Delta_{x_i^{g,k;h}} \in R$ are corrective values, based on which the initial boundaries can be either expanded or narrowed. They can be obtained based on the standard deviation of the values of the x_i variable within the cluster.

Thus, at each generation, the local adaptation of the search area boundaries is performed in order to concentrate computational resources on promising regions of the solution space.

Implementation and discussion

The method was tested on a set of four standard test functions of various dimensions [18]. The effectiveness of the approach is further illustrated using the example of four test functions: Ackley, Beale, Booth and Bukin № 6, as they form a representative sample for the comprehensive evaluation of the effectiveness of optimisation algorithms. The key advantage of this set of functions is that they have the most representative set of properties that pose the greatest complexity for optimisation algorithms.

The Ackley function represents a complex multimodal surface with a large number of local minima, which allows testing the algorithm's ability to avoid premature convergence to suboptimal solutions (8):

$$f(x) = -20 \exp \left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right) - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right) + 20 + e. \quad (8)$$

The Beale function is highly sensitive to the choice of a starting point (9):

$$F(\mathbf{x}) = \sum_{i=1}^d \left[(1.5 - x_{2i-1} + x_{2i-1} x_{2i})^2 + (2.25 - x_{2i-1} + x_{2i-1} x_{2i}^2)^2 + (2.625 - x_{2i-1} + x_{2i-1} x_{2i}^3)^2 \right],$$

$$\mathbf{x} = (x_1, \dots, x_{2d}) \in \mathbb{R}^{2d}. \quad (9)$$

The Booth function, having a relatively simple structure with a single clear-cut global minimum, serves as a test for the basic performance of optimisation methods (10):

$$f(x, y) = (x + 2y - 7)^2 + (2x + y - 5)^2. \quad (10)$$

Thanks to the presence of gaps and characteristic ‘ravines,’ the Bukin № 6 function allows evaluating the stability of algorithms to extreme changes in gradient (11):

$$f(x, y) = 100 \sqrt{(y - 0.01x^2)} + 0.01(x + 10). \quad (11)$$

Fig. 3 shows the contour plots of the above test functions for the case of two variables.

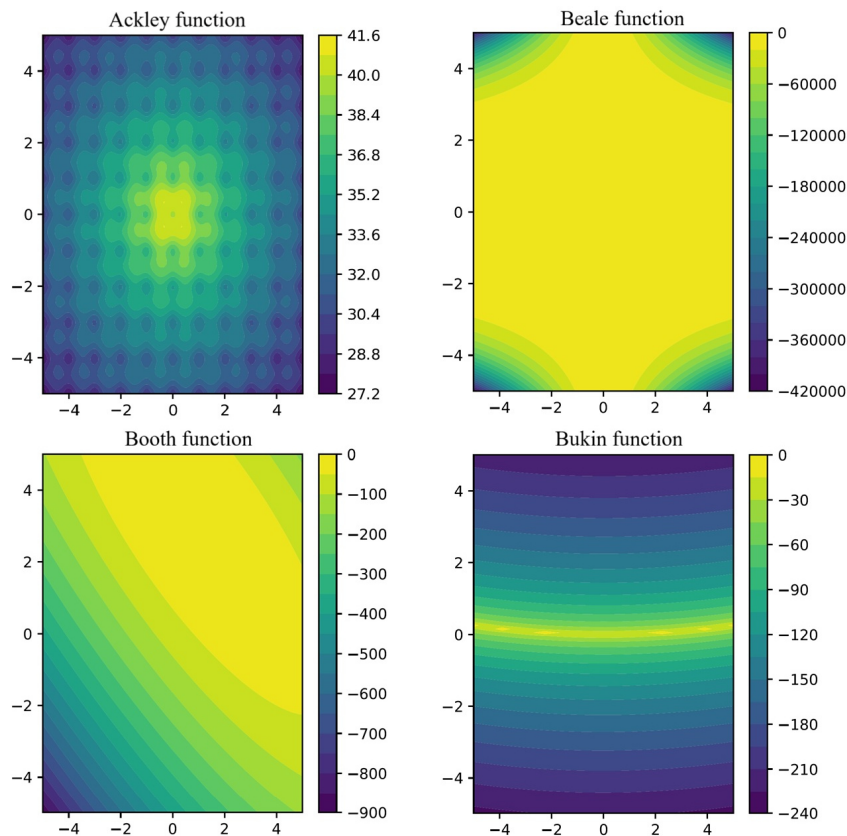


Рис. 3. Изолинии тестовых функций

Fig. 3. Contour plots of test functions

Fig. 4 shows the visualisation of the effectiveness evaluation of the proposed approach using the examples of the above functions for the Ackley, Beale, Booth and Bukin № 6 functions, respectively. The choice is determined by the different nature of these test problems: the Ackley function is traditionally used as a benchmark for testing the scalability of optimisation methods in high-dimensional space; the multidimensional modification of the Beale function allows evaluating the behaviour of the algorithm at medium dimensions with the clear-cut dependence on initial conditions; The Booth and Bukin No. 6 functions are two-dimensional by definition and are used for basic validation of the correctness and stability of the method under conditions of simple (for Booth) and rapidly changing (for Bukin No. 6) destination surface relief. The figure shows the results of applying five clustering methods: k-means, k-medians, DBSCAN, agglomerative clustering and moving average from top to bottom, respectively.

It can be seen from Fig. 4 that for the complex multidimensional functions (Ackley, Beale), the best results are achieved when using 3–6 clusters, where the median values of the number of iterations are lower and the spread is smaller. For the two-dimensional functions (Booth, Bukin №. 6), the influence of the parameter is minimal, thus 1–2 clusters are sufficient without any noticeable loss of quality.

Based on the algorithm runs on the two-dimensional test functions described above, it was discovered that the optimal range of clusters for most tasks was between 3 and 6. The k-means method was used to analyse the multidimensional test functions, ensuring stable point separation and accelerating algorithm convergence.

Table 1 shows the average number of evaluations of objective functions with standard deviation for different dimensions and numbers of clusters. The functions are selected from a standard set of test functions that can be generalised to higher-dimensional spaces.

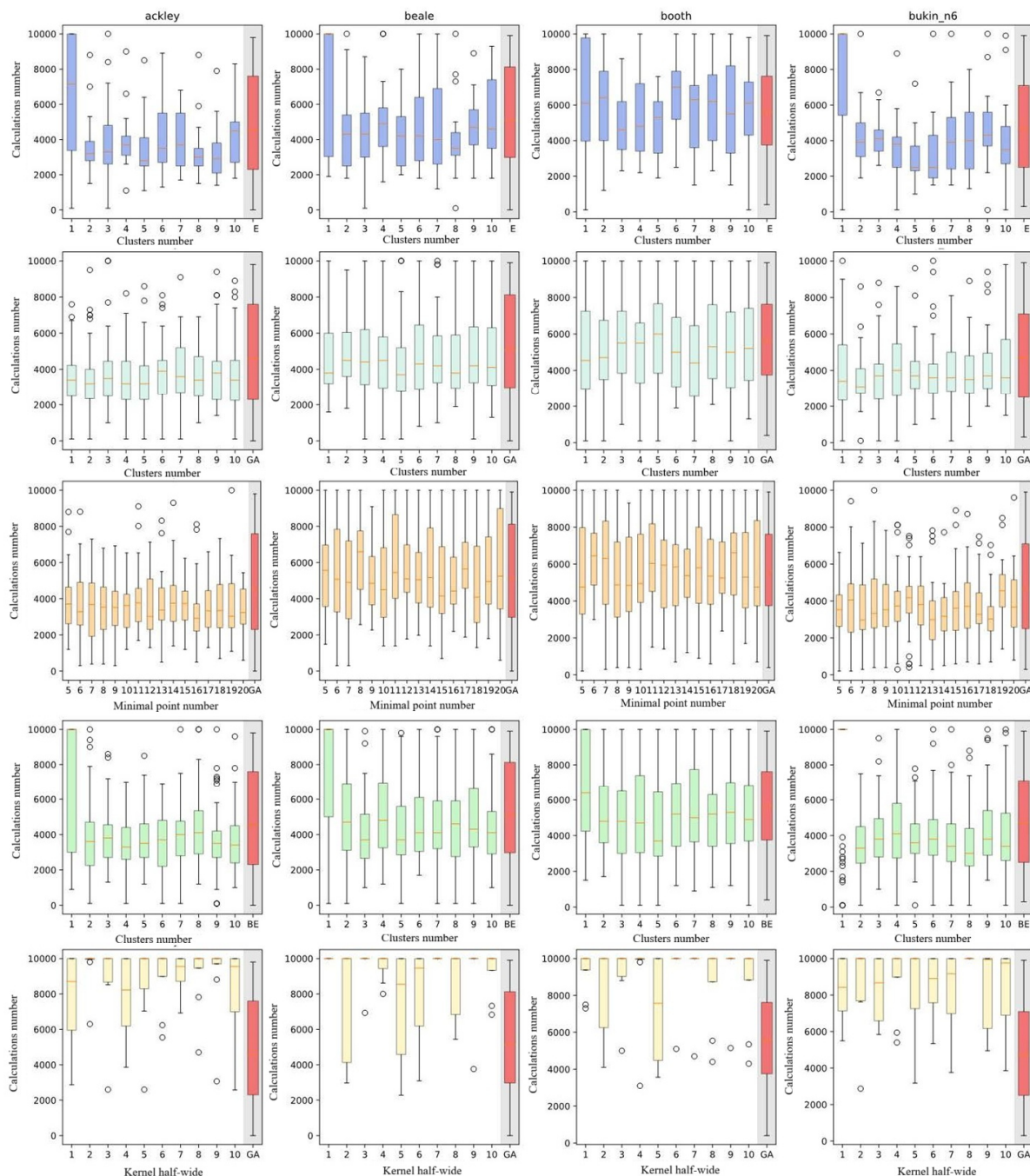


Рис. 4. Диаграммы межквартильного размаха результатов экспериментов. Представлены результаты применения пяти методов кластеризации: k-средних (k-means), k-медианных (k-medians), DBSCAN, агломеративной кластеризации и скользящего среднего (от верхнего к нижнему ряду, соответственно)

Fig. 4. Interquartile (IQR) range diagrams of the experimental results. The figure shows the outcomes of five clustering methods: k-means, k-medians, DBSCAN, agglomerative clustering, and moving average (from top to bottom row, respectively)

Table 1

**Average number of evaluations of an objective function ($\pm$ standard deviation)
for the multidimensional test functions with different dimensions and numbers of clusters**

Function	Dimension	3 clusters	4 clusters	5 clusters	6 clusters
Sphere	10	1187 $\pm$ 132	2295 $\pm$ 214	2478 $\pm$ 289	2221 $\pm$ 367
	20	2644 $\pm$ 305	5548 $\pm$ 412	4822 $\pm$ 497	4133 $\pm$ 721
	50	4758 $\pm$ 623	10187 $\pm$ 807	11643 $\pm$ 1002	11387 $\pm$ 1154
	100	9442 $\pm$ 1189	8921 $\pm$ 1030	8737 $\pm$ 1655	9845 $\pm$ 2033
Rosenbrock	10	3368 $\pm$ 247	3124 $\pm$ 301	4192 $\pm$ 382	4087 $\pm$ 472
	20	5278 $\pm$ 493	4842 $\pm$ 615	5217 $\pm$ 786	6462 $\pm$ 912
	50	8956 $\pm$ 1044	8682 $\pm$ 1218	8447 $\pm$ 1659	8973 $\pm$ 1187
	100	10671 $\pm$ 2047	11348 $\pm$ 1965	10582 $\pm$ 2954	12791 $\pm$ 3092
Rastrigin	10	4142 $\pm$ 198	4084 $\pm$ 153	4226 $\pm$ 433	4211 $\pm$ 498
	20	6637 $\pm$ 527	6438 $\pm$ 642	6582 $\pm$ 804	6276 $\pm$ 986
	50	9672 $\pm$ 1810	9452 $\pm$ 1551	9911 $\pm$ 2792	9374 $\pm$ 2025
	100	16941 $\pm$ 2647	15482 $\pm$ 3082	12653 $\pm$ 3467	13278 $\pm$ 3065
Ackley	10	3885 $\pm$ 107	3851 $\pm$ 286	3682 $\pm$ 154	3718 $\pm$ 449
	20	9154 $\pm$ 554	9247 $\pm$ 718	10368 $\pm$ 601	9012 $\pm$ 1802
	50	15934 $\pm$ 1087	17614 $\pm$ 2762	15761 $\pm$ 2634	16286 $\pm$ 2089
	100	22184 $\pm$ 2794	20378 $\pm$ 3358	19756 $\pm$ 3841	20187 $\pm$ 2712
Stybinsi-Tang	10	4082 $\pm$ 103	3738 $\pm$ 214	4187 $\pm$ 478	3724 $\pm$ 251
	20	6326 $\pm$ 389	6843 $\pm$ 518	7624 $\pm$ 967	6557 $\pm$ 842
	50	10762 $\pm$ 895	10574 $\pm$ 1128	11198 $\pm$ 1376	10867 $\pm$ 1582
	100	20468 $\pm$ 1836	21328 $\pm$ 1215	21642 $\pm$ 2048	18124 $\pm$ 3096

It can be seen from Table 1 that in most cases, the smallest number of evaluations of an objective function is achieved with 3–4 clusters, especially for low dimensions. For high dimensions, the effect of the number of clusters is less clear-cut, but the range of 3–6 clusters remains effective.

The material from the used test functions shows that the correct selection of the k number of clusters can dramatically change the number of generations required to converge to the global optimal solution with the proposed modification. In all the cases, the key factors are:

- complexity of the objective function landscape (presence of local minima and their density, narrow or wide attraction zones);
- adequacy of the choice of k (small k does not allow identifying all ‘hot’ areas, while too large k leads to the overuse of resources);
- The efficiency of the procedure for updating search area boundaries in each cluster ensures that the algorithm quickly shifts its focus to promising regions.

This balance between exploring the solution space and using the results of the study makes the proposed modification converge faster than the standard approach.

In addition, for the comparison of the effectiveness of the GA modification proposed in this article, a series of computational experiments were conducted on a set of CEC 2017 test functions [19]. In the experiments, four combinations of algorithms were considered:

- standard GA without modifications;
- standard GA with the proposed modification;
- algorithm SelfCSHAGA (Self-Configuring Success History-based Adaptation Genetic Algorithm) [20];
- algorithm SelfCSHAGA with the proposed modification.

In the experimental protocol, test functions from the CEC'2017 set were used. For each function, the parameters were selected in order to ensure the maximum possible dimension of the task while maintaining adequate computational complexity and stability of the algorithm. In particular, for most functions, a dimension of 50 variables and a population size of 200 individuals were used. For more computationally complex functions (for example, $f4, f5, f7, f8$), the dimension was reduced to 20 or

10. It should be noted that the f_9 function proved to be sensitive to dimension: when the dimension was increased to 10, the SelfCSHAGA algorithm worked similarly to the standard GA. Nevertheless, with a dimension of 2, the algorithm found the optimum rather quickly.

The number of iterations was also adjusted depending on the complexity of the function and the dimension of the search area: from 100 iterations (for the f_9) to 2000 iterations (for the f_2). Table 2 presents the parameters of the experiments of the standard GA and SelfCSHAGA algorithm on the CEC'2017 test functions. For each function, the task dimension, population size, number of iterations, search boundaries, and discretisation step are indicated. It should be noted that on the functions f_1 , f_2 and f_{10} , the standard GA failed to find the optimum, while the SelfCSHAGA demonstrated successful performance.

Table 2

The parameters of the experiments for the GA and SelfCSHAGA (sSHAGA) on the CEC'2017 functions

Func.	Dimension D (GA)	Dimension D (sSHAGA)	Generations (GA)	Generations (sSHAGA)	Population (GA)	Population (sSHAGA)	Search boundaries	Optimal value
f_1	–	50	–	1100	–	200	$[-100, 100]$	100
f_2	–	50	–	2000	–	200	$[-100, 100]$	200
f_3	20	50	4500	1000	100	200	$[-100, 100]$	300
f_4	10	20	3500	1000	100	200	$[-100, 100]$	400
f_5	10	20	3500	1300	200	200	$[-100, 100]$	500
f_6	20	50	2500	300	100	200	$[-100, 100]$	600
f_7	10	20	2500	500	100	400	$[-50, 50]$	700
f_8	10	10	3500	1100	300	100	$[-50, 50]$	800
f_9	2	2	100	100	100	50	$[-10, 10]$	900
f_{10}	–	10	–	2500	–	200	$[-100, 100]$	1000

Using the same settings, the algorithms with the modifications proposed in this article were tested. A K-means clusterizer with 3 to 6 clusters was used. The analysis of the distribution of the number of objective function evaluations was performed based on the interquartile range. For the standard GA, there is variability in computational efforts between functions (Figs. 5 and 6).

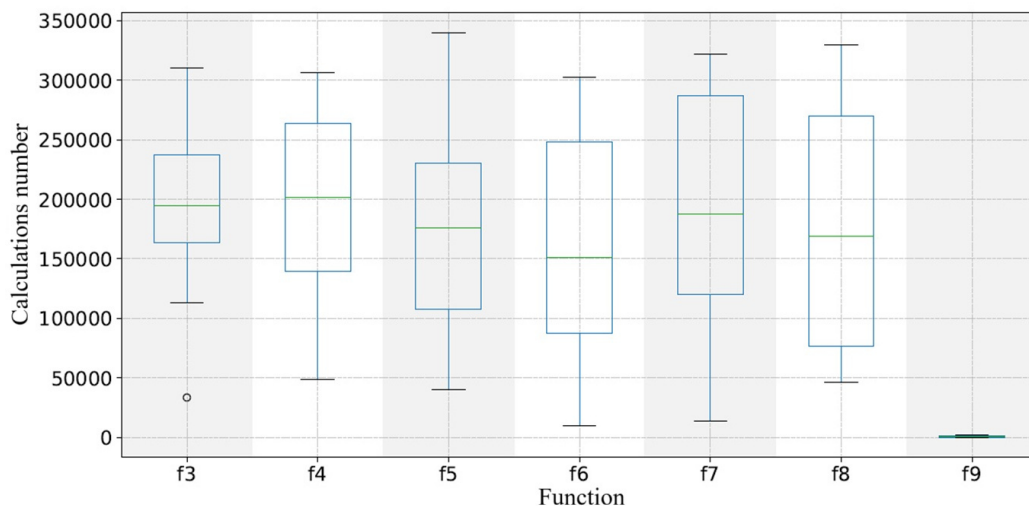


Рис. 5. Диаграммы межквартильного размаха для функций f_3 – f_9 . По оси ординат показано количество вычислений целевой функции. Результаты получены с использованием стандартного генетического алгоритма

Fig. 5. Boxplot diagrams for functions f_3 – f_9 . The y-axis shows the number of objective function evaluations. The results were obtained using the standard genetic algorithm

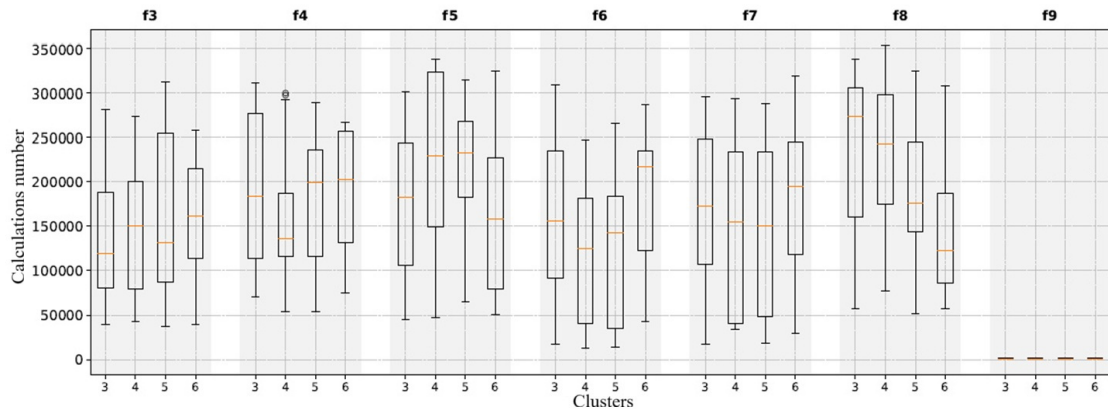


Рис. 6. Диаграммы межквартильного размаха для функций f_3 – f_9 . Каждый столбик соответствует числу кластеров (3, 4, 5, 6), сгруппированных по функциям. По оси ординат показано количество вычислений целевой функции. Результаты получены с использованием стандартного генетического алгоритма с модификацией, предложенной в настоящей диссертации

Fig. 6. Boxplot diagrams for functions f_3 – f_9 . Each bar corresponds to the number of clusters (3, 4, 5, 6), grouped by functions. The y-axis shows the number of objective function evaluations. The results were obtained using the standard genetic algorithm with the modification proposed in this dissertation

For example, the functions f_3 – f_8 demonstrate high median values and interquartile range widths, which indicates scattered consumption of computing resources. The algorithm variant with the modification proposed in the dissertation, including K-means clustering with 3 to 6 clusters, made it possible to reduce the dispersion of the number of evaluations and shift the median towards smaller values. Thus, the generation of individuals within clusters ensured a directed distribution of the search across the solution space, which reduced the number of evaluations required to achieve an approximate optimum.

The analysis of distributing the number of the objective function evaluations for the SelfCSHAGA algorithm (Figs. 7 and 8) shows high stability and low variability compared to the standard GA.

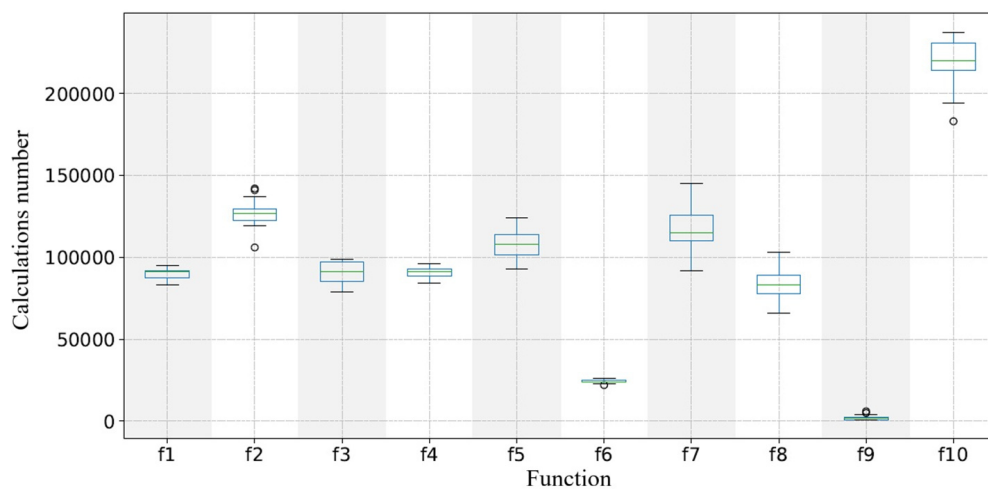


Рис. 7. Диаграммы межквартильного размаха для функций f_1 – f_{10} . По оси ординат показано количество вычислений целевой функции. Результаты получены с использованием алгоритма SelfCSHAGA

Fig. 7. Boxplot diagrams for functions f_1 – f_{10} . The y-axis shows the number of objective function evaluations. The results were obtained using the SelfCSHAGA algorithm

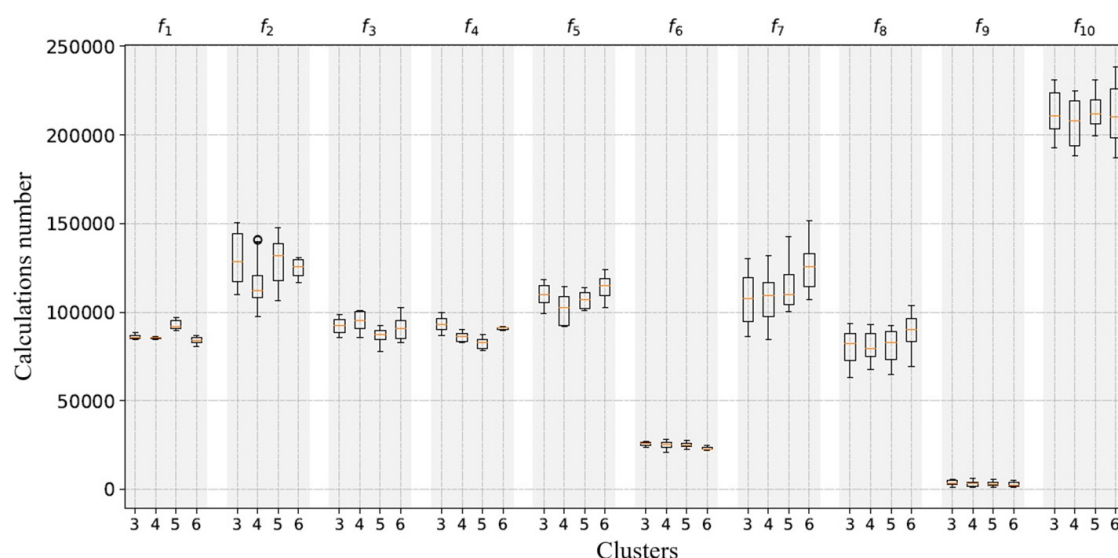


Рис. 8. Диаграммы межквартильного размаха для функций f_1 – f_{10} . Каждый столбик соответствует числу кластеров (3, 4, 5, 6), сгруппированных по функциям. По оси ординат показано количество вычислений целевой функции. Результаты получены с использованием алгоритма SelfCSHAGA с модификацией, предложенной в настоящей статье

Fig. 8. Boxplot diagrams for functions f_1 – f_{10} . Each bar corresponds to the number of clusters (3, 4, 5, 6), grouped by functions. The y-axis shows the number of objective function evaluations. The results were obtained using the SelfCSHAGA algorithm with the modification proposed in this paper

The use of K-means clustering (number of clusters from 3 to 6) further reduces the spread of calculations. By distributing the population across clusters, the algorithm explores solution space more effectively, especially for the functions with high computational complexity (f_3 – f_8), reducing the number of iterations required to reach an approximate optimum and narrowing the interquartile range. For example, for the f_5 and f_7 , the modified algorithm consumes significantly fewer computations than the standard SHAGA, while the IQR is reduced reflecting the more predictable and economical operation of the algorithm.

Conclusion

The study of the proposed approach showed that integrating the method of dynamic correction of the search area based on clustering into GA allows reducing the number of the objective function evaluations significantly due to more accurate localisation of areas with promising solutions. The correction of search area boundaries based on statistics of the distribution of individuals in clusters demonstrates high efficiency in combination with the dynamic redistribution of computational resources.

The numerical experiments on a representative set of test functions confirmed that the effective number of clusters k is determined by the topology of the objective function and can vary over a wide range. Small values of k simplify landscape modelling but increase the risk of missing individual extrema, while an excessively large k leads to an increase in the number of calculations. Choosing an effective value of k (usually 3–6) provides the best balance between exploring the solution space and using the results of the exploration, reducing the number of generations to convergence by sometimes tens of times compared to the standard approach.

These results confirm the relevance and utility of the proposed approach and justify its further development. In particular, it would be advisable to study the possibility of automatic adaptation of k during the optimisation process, as well as to expand the list of clustering methods used by adding the HDBSCAN algorithms [21] and Kohonen self-organising maps [22]. Furthermore, a promising direc-

tion for development is the combination of density and graph approaches for the more flexible control of search boundaries in complex multidimensional problems.

Библиографические ссылки

1. Semenkin E., Semenkina M. Spacecrafts' control systems effective variants choice with self-configuring genetic algorithm // ICINCO 2012 – Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics. Rome, 28–31 July 2012. 2012. Vol. 1. P. 84–93.
2. Hidden-variables genetic algorithm for variable-size design space optimal layout problems with application to aerospace vehicles / J. Gamot et al. // Engineering Applications of Artificial Intelligence. 2023. Vol. 121. P. 105941.
3. Srinivas M., Patnaik L. Adaptive probabilities of crossover and mutation in genetic algorithms // IEEE Transactions on Systems, Man, and Cybernetics. 1994. Vol. 24, No. 4. P. 656–667.
4. Liu C., Liu K. An asynchronous island-model genetic algorithm for large-scale global optimization // Applied Soft Computing. 2019. Vol. 76. P. 637–649.
5. Izzo D., Rucinski M., Ampatzis C. Parallel global optimisation meta-heuristics using an asynchronous island-model // Proc. 2009 IEEE Congress on Evolutionary Computation. 2009. P. 2301–2308.
6. Gendreau M., D'Amours G., Rousseau S. Memetic algorithms: Hybrid genetic algorithms and local search methods for combinatorial optimization // Computers & Operations Research. 2020. Vol. 112. P. 104778.
7. Verma S., Pant M., Snasel V. A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems // IEEE access. 2021. Vol. 9. P. 57757–57791.
8. Adaptive replacement strategies for MOEA/D / Wang Z. et al. // IEEE transactions on cybernetics. 2015. Vol. 46, No. 2. P. 474–486.
9. Multiobjective genetic programming: Reducing bloat using SPEA2 / Bleuler S. et al. // Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546). 2001. Vol. 1. P. 536–543.
10. Jain L., Sinha A. Adaptive reference point selection in NSGA-III for many-objective optimization // Applied Soft Computing. 2021. Vol. 105. P. 107237.
11. Shi Y., Song Y., Sun W. Surrogate-assisted multi-objective genetic algorithm for computationally expensive problems // Applied Soft Computing. 2021. Vol. 111. P. 107637.
12. Galván E., Mooney P. Neuroevolution in deep neural networks: Current trends and future challenges // IEEE Transactions on Artificial Intelligence. 2021. Vol. 2, No. 6. P. 476–493.
13. Papavasileiou E., Cornelis J., Jansen B. A systematic literature review of the successors of “neuroevolution of augmenting topologies” // Evolutionary Computation. 2021. Vol. 29, No. 1. P. 1–73.
14. Katoch S., Chauhan S. S., Kumar V. A review on genetic algorithm: past, present, and future // Multimedia Tools and Applications. 2021. Vol. 80. P. 8091–8126.
15. Alhijawi B., Awajan A. Genetic algorithms: Theory, genetic operators, solutions, and applications // Evolutionary Intelligence. 2024. Vol. 17, No. 3. P. 1245–1256.
16. Sohail A. Genetic algorithms in the fields of artificial intelligence and data sciences // Annals of Data Science. 2023. Vol. 10, No. 4. P. 1007–1018.
17. Schubert E. Stop using the elbow criterion for k-means and how to choose the number of clusters instead // ACM SIGKDD Explorations Newsletter. 2023. Vol. 25, No. 1. P. 36–42.
18. Jamil M., Yang X. S., Zepernick H. J. Test functions for global optimization: a comprehensive survey // Swarm intelligence and Bio-inspired Computation. 2013. P. 193–222.
19. Wu G., Mallipeddi R., Suganthan P. N. Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization // National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report. 2017. Vol. 9. P. 2017.

20. Шерстнев П. А., Семенкин Е. С. SelfCSHAGA: Самоконфигурируемый генетический алгоритм оптимизации с адаптацией на основе истории успеха // Вестник МГТУ им. Н. Э. Баумана. Сер. Приборостроение. 2025. № 2 (151). С. 122–139.
21. Stewart G., Al-Khassaweneh M. An implementation of the HDBSCAN* clustering algorithm // Applied Sciences. 2022. Vol. 12, No. 5. P. 2405.
22. Gorokhovatskyi V. et al. Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set. IEEE Access, 2024.

References

1. Semenkin E., Semenkina M. Spacecrafts' control systems effective variants choice with self-configuring genetic algorithm // ICINCO 2012 – Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics. Rome, 28–31 July 2012. 2012, Vol. 1, P. 84–93.
2. Gamot J. et al. Hidden-variables genetic algorithm for variable-size design space optimal layout problems with application to aerospace vehicles. Engineering Applications of Artificial Intelligence. 2023, Vol. 121, P. 105941.
3. Srinivas M., Patnaik L. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*. 1994, Vol. 24, No. 4, P. 656–667.
4. Liu C., Liu K. An asynchronous island-model genetic algorithm for large-scale global optimization. *Applied Soft Computing*, 2019, Vol. 76, P. 637–649.
5. Izzo D., Rucinski M., Ampatzis C. Parallel global optimisation meta-heuristics using an asynchronous island-model. *Proc. 2009 IEEE Congress on Evolutionary Computation*, IEEE, 2009, P. 2301–2308.
6. Gendreau M., D'Amours G., Rousseau S. Memetic algorithms: Hybrid genetic algorithms and local search methods for combinatorial optimization. *Computers & Operations Research*, 2020, Vol. 112, P. 104778.
7. Verma S., Pant M., Snasel V. A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems. *IEEE access*. 2021, Vol. 9, P. 57757–57791.
8. Wang Z. et al. Adaptive replacement strategies for MOEA/D. *IEEE transactions on cybernetics*. 2015, Vol. 46, No. 2, P. 474–486.
9. Bleuler S. et al. Multiobjective genetic programming: Reducing bloat using SPEA2. *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*. IEEE, 2001, Vol. 1, P. 536–543.
10. Jain L., Sinha A. Adaptive reference point selection in NSGA-III for many-objective optimization. *Applied Soft Computing*. 2021, Vol. 105, P. 107237.
11. Shi Y., Song Y., Sun W. Surrogate-assisted multi-objective genetic algorithm for computationally expensive problems. *Applied Soft Computing*. 2021, Vol. 111, P. 107637.
12. Galván E., Mooney P. Neuroevolution in deep neural networks: Current trends and future challenges. *IEEE Transactions on Artificial Intelligence*. 2021, Vol. 2, No. 6, P. 476–493.
13. Papavasileiou E., Cornelis J., Jansen B. A systematic literature review of the successors of “neuroevolution of augmenting topologies”. *Evolutionary Computation*. 2021, Vol. 29, No. 1, P. 1–73.
14. Katoch S., Chauhan S. S., Kumar V. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*. 2021, Vol. 80, P. 8091–8126.
15. Alhijawi B., Awajan A. Genetic algorithms: Theory, genetic operators, solutions, and applications. *Evolutionary Intelligence*. 2024, Vol. 17, No. 3, P. 1245–1256.
16. Sohail A. Genetic algorithms in the fields of artificial intelligence and data sciences. *Annals of Data Science*. 2023, Vol. 10, No. 4, P. 1007–1018.
17. Schubert E. Stop using the elbow criterion for k-means and how to choose the number of clusters instead. *ACM SIGKDD Explorations Newsletter*. 2023, Vol. 25, No. 1, P. 36–42.
18. Jamil M., Yang X. S., Zepernick H. J. Test functions for global optimization: a comprehensive survey. *Swarm intelligence and Bio-inspired Computation*. 2013, P. 193–222.

19. Wu G., Mallipeddi R., Suganthan P. N. Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*. 2017, Vol. 9, P. 2017.
20. Sherstnev P. A., Semekin E. S. [SelfCSHAGA: Self-configuring genetic optimization algorithm with adaptation based on success history]. *Vestnik MGTU im. N. E. Baumana. Ser. Priborostroenie*. 2025, No. 2 (151), P. 122–139 (In Russ.).
21. Stewart G., Al-Khassaweneh M. An implementation of the HDBSCAN* clustering algorithm. *Applied Sciences*. 2022, Vol. 12, No. 5, P. 2405.
22. Gorokhovatskyi V. et al. Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set. *IEEE Access*, 2024.

© Malashin I. P., Tynchenko V. S., 2025

Малашин Иван Павлович – инженер-программист; Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет). E-mail: ivan.p.malashin@gmail.ru. <https://orcid.org/0009-0008-8986-402X>.

Тынченко Вадим Сергеевич – доктор технических наук, доцент; Сибирский государственный университет науки и технологий имени академика М. Ф. Решетнева. E-mail: vadimond@mail.ru. <https://orcid.org/0000-0002-3959-2969>.

Malashin Ivan Pavlovich – Software engineer; Bauman Moscow State Technical University (National Research University). E-mail: ivan.p.malashin@gmail.ru. <https://orcid.org/0009-0008-8986-402X>.

Tynchenko Vadim Sergeevich – Dr. Sc., Associate Professor; Reshetnev Siberian State University of Science and Technology. E-mail: vadimond@mail.ru. <https://orcid.org/0000-0002-3959-2969>.

Статья поступила в редакцию 19.06.2025; принята к публикации 12.09.2025; опубликована 13.10.2025
The article was submitted 19.06.2025; accepted for publication 12.09.2025; published 13.10.2025

Статья доступна по лицензии Creative Commons Attribution 4.0
The article can be used under the Creative Commons Attribution 4.0 License