ISSN 2786-7102 (Online), ISSN 2786-7633 (Print)
https://doi.org/10.15407/srenergy2025.03.065

UDC 517.9:519.6

Vladyslav Khaidurov[1,2]*, PhD (Engin.), Senior Researcher, https://orcid.org/0000-0002-4805-8880
Artem Zinchenko[1], PhD (Engin.), Associate Professor, https://orcid.org/0000-0003-1586-3645
Tamara Tsiupii[3], PhD (Phys. & Math.), Associate Professor, https://orcid.org/0000-0003-2206-2897
Roman Yarovoy[4], PhD (Engin.), https://orcid.org/0000-0001-8978-8137
[1]National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37, Beresteiskyi Prosp., Kyiv, 03056, Ukraine;
[2]General Energy Institute of NAS of Ukraine, 172, Antonovycha St., Kyiv, 03150, Ukraine;
[3]National University of Life and Environmental Sciences of Ukraine, 15, Heroiv Oborony St., Kyiv, 03041, Ukraine;
[4]European University, 16 V, Academician Vernadsky Blvd., Kyiv, 03115, Ukraine
*Corresponding author: allif0111@gmail.com

MODIFICATION OF SWARM INTELLIGENCE METHODS FOR OPTIMIZATION OF COMPLEX PROCESSES, OBJECTS, AND SYSTEMS

Abstract. *The development of high-speed methods and algorithms for global multidimensional optimization and their modifications in various fields of science, technology, and economics is an urgent problems that involves reducing computing costs, accelerating and effectively finding solutions to such problems. Due to the fact that most serious problems involve the search for tens, hundreds or thousands of optimal parameters of mathematical models, the search space for these parameters grows non-linearly. Modern swarm intelligence has significant potential for application in the energy industry due to its ability to optimize and solve complex problems. With its help, it is possible to solve scientific and applied problems of optimizing energy consumption in buildings, industrial complexes and urban systems, reducing energy losses and increasing the efficiency of resource use, as well as for the construction of various elements of energy systems in general. Well-known methods and algorithms of swarm intelligence are also actively used to forecast energy production from renewable sources, such as solar and wind energy. This allows better management of energy sources and planning of their use. The relevance of modifications of methods and algorithms is due to the issues of speeding up their work when solving machine learning problems, in particular, in nonlinear regression models, classification, clustering problems, where the number of observed data can reach tens and hundreds of thousands or more. The work considers and modifies well-known effective methods and algorithms of swarm intelligence (particle swarm optimization algorithm, bee optimization algorithm, differential evolution method) for finding solutions to multidimensional extremal problems with and without restrictions, as well as problems of nonlinear regression analysis. The effectiveness of the modified methods on various classical and applied problems, which are used in the design of elements of complex objects and their systems, is demonstrated. A comparative analysis of the results of these methods was carried out.*
Keywords: optimization, swarm intelligence, mathematical modelling, nonlinear regression, complex objects and systems.

1. Introduction

With the development of modern computing systems, its application for solving optimization problems, there is a need for the design of complex systems in thermal power engineering [1–3], mathematical modeling. The modern theory of optimization methods, taking into account its new applications, has undergone significant changes, which consist in the development of new and modification of existing methods and algorithms for finding solutions to optimization, inverse problems and problems in incorrect formulation [4].

Considerable interest is now focused on algorithms and methods of swarm intelligence, which are finding more and more applied applications in modern science and technology. Swarm intelligence is a concept

that is inspired by observations of the behavior of animal colonies in nature, such as ants, bees, and swarms of birds. This approach to artificial intelligence and optimization is based on modeling the behavior of individual agents that interact with each other and with the environment. The collective behavior of these agents is actively used to solve complex problems and find optimal solutions without centralized management.

The applications of swarm intelligence in science and technology include various fields such as optimization, robotics, data networks, and even machine learning. For example, optimization algorithms based on swarm intelligence can be used to solve routing problems, find optimal solutions in complex parameter spaces, or manage distributed systems [1, 5–8].

In the field of robotics and the management of groups of robots, swarm intelligence allows for the creation of efficient algorithms to coordinate the actions of many robots without centralized control. This is especially useful in scenarios where adaptability to changing environmental conditions is required. In data networks, swarm algorithms can be used to optimize routing and traffic management, taking into account dynamic network conditions and changes in load. In the field of machine learning, swarm methods and algorithms can be applied to create effective deep learning methods, in particular, in the problems of optimizing loss functions and finding hyperparameters [1].

2. Methods and materials

2.1. Problem statement

Most of the applied optimization problems are reduced to finding the global extrema of functions in the classical formulation, which is presented as follows [1, 8]:

$$y = F(\mathbf{X}) \to \min(\max), \quad \mathbf{X} \in G,\ G \subset \mathbb{R}^n,$$

where $\mathbf{X} = (X_1, X_2, \cdots, X_n)$, G is the search space for the values of the arguments **X**, n is the dimension of the search space of the global extremum of the function. X_j, $j = \overline{1;n}$ – factor (independent) variables that establish a causal relationship with the dependent (resultant) variable y. The function $F(\mathbf{X})$ is often subject to functional constraints, which are presented in the form of the following functional dependencies:

$$g_i(\mathbf{X}) \le S_i,\ i = \overline{1;K}, \quad g_i(\mathbf{X}) = S_i,\ i = \overline{K+1;R}.$$

The complexity of the objective function $F(\mathbf{X})$ and the corresponding constraints $g_i(\mathbf{X})$ is determined by a specific technical problems [1, 9]. The appearance *of* $F(\mathbf{X})$ and $g_i(\mathbf{X})$ determines the choice of the optimization method that will be used to search for $\mathbf{X}^* = (X_1^*, X_2^*, \cdots, X_n^*)$, $F(\mathbf{X}^*)$.

It should be noted that the dependence of the species: $y = F(\mathbf{X}) \to \min$.

Such functional dependencies arise in regression (linear and nonlinear) analysis, in the problems of classification, data clustering, in the construction of effective mathematical models of process control, forecasting of time series, modernization of objects and systems, as well as in the development of complex energy complexes in today's conditions.

In this paper, we will consider the problems of searching for a global minimum of problems without functional constraints on the objective function / functional, problems with constraints in the form of functions, as well as problems of searching for nonlinear regression parameters, as well as methods for their search [1, 2, 9]. In this case, further research will focus on methods and algorithms that have a stochastic component. Three well-known methods and algorithms were taken as a basis: the particle swarm optimization algorithm, the bee optimization algorithm, and the differential evolution method. The criteria for selecting basic methods and algorithms are based on the following well-known facts:

- the chosen methods and algorithms are relatively easy to understand, making them accessible to a wide range of software developers and engineers;
- the chosen methods and algorithms are able to adapt to changes in the environment or optimization problems by making changes to the parameters or search strategy.
- the selected methods and algorithms are easily parallelized to perform simultaneous data processing;

– the chosen methods and algorithms do not use any information about the derivatives of the objective function and the corresponding constraints on it;
– there are comparatively few parameters in the selected methods and algorithms;
– the chosen methods and algorithms are very effective for finding the global extremum of a function.

2.2. Problem solving methods

2.2.1. PSO algorithm

PSO uses a swarm of particles, where each particle represents a potential solution to a problem. Initially, all the particles of the swarm occupy a random position in the space of the search for the solution of the problem and have small random velocities. In the final iterations, the set of particles converges to one or more optimums that are global (if there are several rather than one). The behavior of a particle in the solution-seeking hyperspace is constantly adjusting to its experience and that of its neighbors. In addition, each particle remembers its best position with the achieved local best value of the objective (fitness) function and knows the best position of the particles of its neighbors, where the global optimum of the function was reached at the moment. In the search process, the swarm particles exchange information about the best results achieved and change their positions and speeds according to certain rules based on the currently available information about local and global achievements. In this case, the global best result is known to all particles and it is corrected in the case when some particle of the swarm finds a better position with a result that exceeds the current global optimum. Each *i*-th particle is characterized by an iteration n of its position $x_i(n)$ in hyperspace and its velocity of motion $v_i(n)$. The velocity of the *i*-th particle is calculated as

$$v_i(n+1) = v_i(n) + \alpha_1\left(x_i^{best}(n) - x_i(n)\right)r_1 + \alpha_2\left(x^*(n) - x_i(n)\right)r_2, \quad (1)$$

The position of the *i*-th particle is calculated as

$$x_i(n+1) = x_i(n) + v_i(n+1), \quad (2)$$

where $x_i(n) = \left(x_{i1}(n); x_{i2}(n); \ldots; x_{iM}(n)\right)$ – the position of the *i*-th particle in the iteration n; $x_i^{best}(n) = \left(x_{i1}^{best}(n); x_{i2}^{best}(n); \ldots; x_{iM}^{best}(n)\right)$ – the best position of the *i*-th particle (personal best position); $x^*(n) = \left(x_1^*(n); x_2^*(n); \ldots; x_M^*(n)\right)$ – the best position for the entire population (global best position); $v_i(n) = \left(v_{i1}(n); v_{i2}(n), \ldots; v_{iM}(n)\right)$ is the velocity vector of the *i*-th particle in the iteration n; α_1, α_2 – positive acceleration coefficients that regulate the contribution of the cognitive and social components; $r_1 = \left(r_{11}; r_{12}; \ldots; r_{1M}\right)$, $r_1 = \left(r_{21}; r_{22}; \ldots; r_{2M}\right)$ are random number vectors that introduce an element of randomness into the search process.

Let us consider the influence of various constituents in calculating the velocity of a particle according to (1). The first term in (1) $v_i(n)$ preserves the previous direction of the velocity of the *i*-th parts and can be considered as the moment that prevents a sharp change in the direction of the velocity and acts as an inertia velocity. Cognitive velocity $\alpha_1\left(x_i^{best}(n) - x_i(n)\right)r_1$ determines the characteristics of a particle with respect to its prehistory, which maintains a better position for a given particle. The effect of this term is that it tries to bring the particle back to a better achieved position. The third term $\alpha_2\left(x^*(n) - x_i(n)\right)r_2$ defines the social velocity, which characterizes the particle in relation to its neighbors. The effect of the social component is that it tries to direct each particle towards the global optimum achieved by the swarm (or some of its immediate environment). The displacement of the particle's position is carried out on the basis of (2).

Algorithm for Optimizing Numerical Functions

1. Initializing

1.1. Specifying Parameters α_1, α_2, and $\alpha_1, \alpha_2 \in (0; 4)$.

1.2. Set the maximum number of iterations N, population size K, the length of the particle position vector M minimum and maximum values of the position vector $x_j^{min}, x_j^{max}, j \in \overline{1,M}$, minimum and maximum values for the velocity vector $v_j^{min}, v_j^{max}, j \in \overline{1,M}$, And $v_j^{max} > 0$.

1.3. Defining the Cost Function (Goal Function) $F(\boldsymbol{x}) \to min,\ \boldsymbol{x} = (x_1, \ldots, x_M)$,
where $\boldsymbol{x}$ is the position vector of the particle.

1.4. Creating the Original Population P

1.4.1. Particle Number $k = 1, P = \varnothing$

1.4.2. Randomly create a vector position x_k

$$\boldsymbol{x_k} = (x_{k1}, \ldots, x_{kM}),\ x_{kj} = x_j^{min} + \left(x_j^{max} - x_j^{min}\right) rand(),$$

where $rand()$ is a function that returns a uniformly distributed random number in the range [0; 1]

1.4.3. Creating a Better Position Vector x_k^{best} : $x_k^{best} = x_k$

1.4.4. Randomly Generate a Velocity Vector v_k

$$v_k = (v_{k1}, \ldots, v_{kM}),\ v_{kj} = v_j^{min} + \left(v_j^{max} - v_j^{min}\right) rand()$$

or

$$v_{kj} = 0$$

1.4.5. If $\left(x_k, x_k^{best}, v_k\right) \notin P$, then $P = P \cup \left\{\left(x_k, x_k^{best}, v_k\right)\right\}, k = k + 1$

1.4.6. If $k \le K$, then go to step 1.4.2

1.4.7. Define a particle <u>of the current population</u> with the best position

$$k^* = \underset{k}{\operatorname{argmin}} F(x_k),\ x^* = x_{k^*}$$

2. Iteration Number $n = 1$
3. Particle Number $k = 1$
4. Velocity vector modification

4.1. $r_1 = rand(),\ r_2 = rand()$

4.2. $v_k = v_k + \alpha_1 \left(x_k^{best} - x_k\right) r_1 + \alpha_2 \left(x^* - x_k\right) r_2$

4.3. Speed limits v_k present, i.e.

$$v_{kj} = \max\left\{v_j^{min}, v_{kj}\right\},\ v_{kj} = \min\left\{v_j^{max}, v_{kj}\right\},\ j \in \overline{1,M}$$

or absent

5. Position Modification

5.1. $\boldsymbol{x_k} = \boldsymbol{x_k} + \boldsymbol{v_k}$

5.2. $j = 1$

5.3. If $x_{kj} < x_j^{min}$, then $x_{kj} = x_j^{min} + \left|x_{kj} - x_j^{min}\right|, v_{kj} = -v_{kj}$

5.4. If $x_{kj} > x_j^{max}$, then $x_{kj} = x_j^{max} - \left|x_{kj} - x_j^{max}\right|, v_{kj} = -v_{kj}$

5.5. If $j < M$, then $j = j + 1$, Go to step 5.3

6. Definition <u>personal (local) best</u> Position:
if $F(x_k) \le F\left(x_k^{best}\right)$, then $x_k^{best} = x_k$
7. If $k < K$, then $k = k + 1$
8. Determine the particle of the current population that is best in terms of the function of the target

$$k^* = \underset{k}{\operatorname{argmin}} F(x_k)$$

9. Determining the Global Best Position:

   if $F\left(x_{k^*}\right) < F\left(x^*\right)$, then $x^* = x_{k^*}$

10. Stop condition:

    If $n < N$, then $n = n+1$, go to step 3

The result is *x**.

2.2.2. Bees algorithm

The bee algorithm is based on the behavior of honey bees. It is based on the behavior of foraging bees and is an extension of the bee system. There is a phase of the worker bee (busy foraging) and the scout bee. The purpose of the algorithm is to determine the location of good areas in the search space. Scout bees perform a random search. The found good (in terms of the value of the objective function / functionality) areas are investigated with the help of local search. The solution corresponds to the position of the bee located in a certain area.

Algorithm for Optimizing Numerical Functions

1. Initializing
   - 1.1. Specifying Parameters $\delta, \eta_{max}, \alpha$ to create a circle, and $\delta \in (0;1)$, $\eta_{max} \in (0;1)$, $\alpha \in (0;1)$.
   - 1.2. Set the maximum number of iterations N, population size K, the number of plots (and the bees of these plots) L_s, number of elite plots L_{es}, The number of bee departures in an elite area Z_e, The number of departures of the worker bee in a regular area Z_o, Length of Bee Position Vector M (dimension of the search space), minimum and maximum values for the position vector $x_j^{min}, x_j^{max}, j \in \overline{1, M}$.
   - 1.3. Defining the Cost Function (Goal Function) $F(\boldsymbol{x}) \to min$, $\boldsymbol{x} = (x_1, \ldots, x_M)$, where $\boldsymbol{x}$ is the vector of the bee's position.
   - 1.4. Randomly create a vector of a better position
     $$x^* = \left(x_1^*, x_2^*, \ldots, x_M^*\right), \; x_{kj} = x_j^{min} + \left(x_j^{max} - x_j^{min}\right) rand(),$$
     where $rand()$ is a function that returns a uniformly distributed random number in the range $[0;1]$
   - 1.5. Creating the Original Population
     - 1.5.1. Bee Number $k = 1, P = \varnothing$
     - 1.5.2. Randomly create a vector position x_k
       $$\boldsymbol{x_k} = \left(x_{k1}, \ldots, x_{kM}\right), \; x_{kj} = x_j^{min} + \left(x_j^{max} - x_j^{min}\right) rand()$$
     - 1.5.3. If $x_k \notin P$, then $P = P \cup \{x_k\}, k = k+1$
     - 1.5.4. If $k \le K$, then go to step 1.5.2
2. Iteration Number $n = 1$
3. Identify the bee with the best position $k^* = \underset{k}{\operatorname{argmin}} F\left(x_k\right)$
3. Particle Number $k = 1$
4. If $F\left(x_{k^*}\right) \le F\left(x^*\right)$, then $x^* = x_{k^*}$
5. Arrange P by the value of the objective function, i.e. $F\left(\boldsymbol{x_k}\right) < F\left(\boldsymbol{x_{k+1}}\right)$
6. Worker Bee Phase (Local Search)
   - 6.1. Plot number $l = 1$
   - 6.2. Determine the size of the circle

$$Z = \begin{cases} Z_e, 1 \le l \le L_{es} \\ Z_o, L_{es} < l \le L_s \end{cases}$$

6.3. Create a Circle for a Position *l*-(a) To the extent permitted

6.3.1. $\eta(n) = \eta_{max}\alpha^n$

6.3.2. $x_{zj} = x_{lj} + \eta(n)\delta\left(x_j^{max} - x_j^{min}\right)\left(-1 + 2rand(\,)\right),\ \ j \in \overline{1;M}, z \in \overline{1;Z}$

6.3.3. $x_{zj} = \max\left\{x_j^{min}; x_{zj}\right\},\ x_{zj} = \min\left\{x_j^{max}; x_{zj}\right\},\ \ j \in \overline{1;M}, z \in \overline{1;Z}$

6.4. $z^* = \underset{z}{\operatorname{argmin}} F\left(x_z\right)$

6.5. *Якщо* $f\left(x_{x_{z^*}}\right) < f\left(x_l\right)$ then $x_l = x_{x_{z^*}}$.

6.6. If $l < L_s$, then $l = l + 1$, Go to step 6.2

7. Scout Bee Phase (Random Search): $x_{lj} = x_j^{min} + \left(x_j^{max} - x_j^{min}\right) rand(\,),\ \ j \in \overline{1;M}, l \in \overline{L_s + 1;K}$
8. Stop Condition
   If $n < N$, then $n = n + 1$, go to step 3
9. Identify the bee with the best position $k^* = \underset{k}{\operatorname{argmin}} F\left(x_k\right),\ \ \boldsymbol{x}^* = \boldsymbol{x}_{\boldsymbol{k}^*}$

The result is *x**.

2.2.3. Differential evolution method

The main idea of the method of differential evolution is the combination of mutation and crossing to efficiently find optimal solutions in multidimensional parameter spaces. The method uses some ideas of genetic algorithms, but, unlike the latter, does not involve working with binary code.

Problem formulation: $F(X) \to \min,\ X = \left(X_1,\ X_2, \ldots, X_M\right)$.

The main stages of the method are as follows:

1) For each chromosome in a population $\overline{x_i}$, $\overline{x_i} = \left(\overline{x_{i1}}, \ldots, \overline{x_{iM}}\right), i = \overline{1;N}$ (N is the dimension of the population), three other random members of that population are selected x_1, x_2, x_3, $\overline{x_i} \ne x_1, \overline{x_i} \ne x_2, \overline{x_i} \ne x_3,\ x_1 \ne x_2 \ne x_3$.
2) A mutant vector is generated: $v = x_1 + F\left(x_2 - x_3\right),\ F \in (0;2)$
3) The vector difference $x_2 - x_3$ is scaled by a user-defined hyperparameter F, $F \in (0;2)$.

A visual illustration of the method is shown in Figure 1.

4) We form a test vector based on crossing $\overline{x_i}$ with a mutant vector v: for each coordinate of the chromosome, a number is generated $r \in (0;1)$ according to the normal distribution.
   - If $r < P$, P is a given constant (another parameter of the algorithm $P \in (0;1)$), then the corresponding coordinate of the chromosome is replaced by $v_j = \overline{x_{ij}}$, $j \in \overline{1;M}$, M – the dimension of the search space (the number of independent variables of the fitness function).

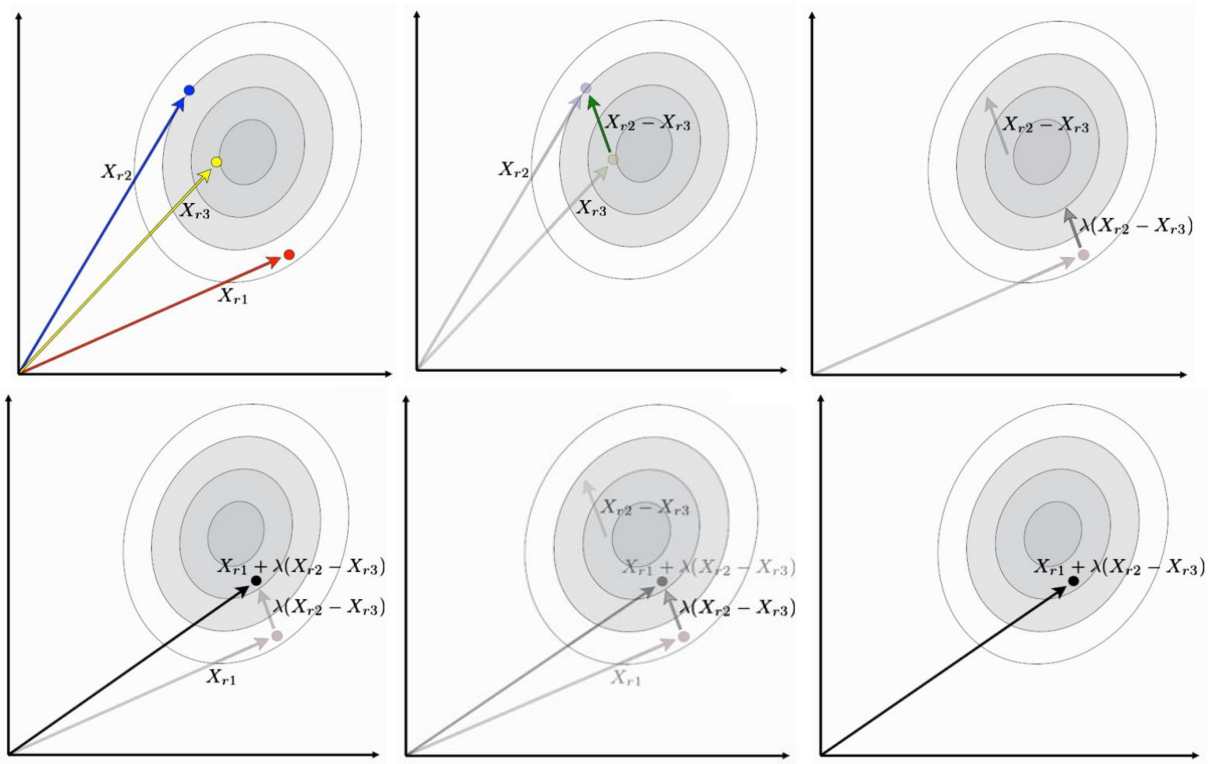


Fig. 1. Illustrative illustration of the method of differential evolution

5) If $f\left(\overline{x_i}\right) > f\left(v\right), \overline{x_i} = v$.

All of the above steps are performed up to the method stop criterion (classical, as in other algorithms and swarm intelligence methods).

2.3. Combination of deterministic and stochastic methods

In complex scientific and technical applied problems, the objective function or functional can be computed for a relatively long time, even on modern computing systems. Stochastic methods and algorithms of global optimization can find global extremes of objective functions without requiring additional information about them, in particular, the differentiability of objective functions. In this case, the total number of calculations increases. This means that the processor time of searching for the global extremum of the objective function also increases. In this case, methods and algorithms are developed that provide for a deterministic component, as well as a stochastic component. The deterministic component works quickly and efficiently refines the solution found, and the stochastic component prevents the method / algorithm from getting stuck in local extremums.

3. Practical results

3.1. Modification of the described methods and algorithms of swarm intelligence

In computational mathematics, each modification of an optimization method/algorithm involves a robot in several ways:

a) reducing the total number of iterations to achieve a certain error. The positive effect of reducing the total number of iterations leads to a decrease in the total processor time for solving a specific problem or solving a specific problem. This, in turn, reduces the overall load on the system;

b) improving the accuracy of calculations. The positive effect of this is to accelerate the convergence of the optimization method/algorithm. This, in turn, again leads to a decrease in the number of iterations, which means a decrease in the processor time required to solve a problem or a specific task.

This paper proposes several modifications of the algorithms described above.

1) Modification of the position of the worst element of the population in the algorithms and methods of swarm intelligence described above. Replace the worst element of x_j^{worst} the population with the average position of the element throughout the population $x_j^{Average} \quad \forall j = \overline{1;M}$ according to the following formula:

$$x_j^{Average} = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \; \forall j = \overline{1;M}, \tag{3}$$

where *N* is the total number of items in the population, and *M* is the dimension of the search space. This approach can be performed on a per-iteration or periodically every *K* iterations.

2) Modification of the position of the elements of the population to the best in increments h. The renewal of each element of the population is carried out according to the formula:

$$x_{ij} = x_{ij} + h \cdot \frac{x_{Best,j} - x_{ij}}{\left\| x_{Best} - x_i \right\|}, \; x_{Best} \neq x_i, \; i = \overline{1;N}, \; j = \overline{1;M}, \tag{4}$$

where x_{Best} is the best vector of the population in terms of the value of the objective function. It should be noted that the step of motion h can be determined proportionally (according to the linear law) to the circumference of the population, which will coincide to the global optimum over time.

3) Changing the hyperparameters of optimization methods/algorithms. This approach involves changing one or more hyperparameters during the operation of the program. For example, in a differential evolution algorithm, the parameter F can be replaced by a random real number in the range from 0 to 2 through *K* iterations.

In the bee algorithm, there is a dependence of the parameter change $\eta(n)$ on the number of iterations *n*. In this case, in order to optimize functions of large dimensions, a problem arises, which is to reduce this parameter in advance. In order to prevent this from happening before finding the global extremum of the function or functional, such a case is assumed to be

$$\eta(n) = \eta_{max}\alpha^{[n/K]}, \tag{5}$$

where [·] is the integer part of the number, *K* is the period that determines how many iterations the value of the η parameter will be updated. In the method of differential evolution, it is proposed to take the hyperparameter *F* as a random number from 0 to 2 every *K* iterations. This is due to the fact that at each iteration, the method finds values closer to the global optimum of the function or functional.

3.2. The structure of the developed software

In the course of the study, a software package for optimization problems was developed, which includes a module of the main wreath of transition to the modules for solving the optimization problems considered in the work, 6 modules (3 modules for solving problems by conventional methods / algorithms and 3 modules with modifications of the corresponding methods / algorithms).

3.3. Functions for testing methods and algorithms and their modifications

The first test function is the function, which looks like this:

$$f(X) = An + \sum_{i=1}^{n}\left(x_i^2 - A\cos(2\pi x_i)\right), \tag{6}$$

where $A = 10, x_i \in [-5,12;5,12], i = \overline{1,n}.$ is the Global Low at $X^* = (x_1^*; x_2^* \ldots; x_n^*) = (0;\ldots;0). \quad f(X^*) = 0.$

The second test function is the Rosenbrock function with functional limitations). Rosenbrock function bounded by a cubic curve and a straight line.

$$\begin{aligned} &f(x,y) = (1-x)^2 + 100\left(y - x^2\right)^2 \to \min, \\ &(x-1)^3 - y + 1 < 0; \; x + y - 2 < 0, \; x \in [-1,5;1,5], y \in [-0,5;2,5]. \end{aligned} \tag{7}$$

Optimal value: $f_{min}(1;1)=0.$

The third test function is the Rosenbrock function, bounded by the disk. It looks like this:

$$f(x,y)=(1-x)^2+100\left(y-x^2\right)^2\to\min,$$
$$x^2+y^2<2,\ x\in[-1,5;1,5], y\in[-1,5;2,5]. \tag{8}$$

Optimal value: $f_{min}(1;1)=0.$

The fourth Mishra-Bird test function with functional limitations, which looks like this:

$$f(x,y)=e^{(1-\cos x)^2}\sin y+e^{(1-\sin y)^2}\cos x+(x-y)^2\to\min,$$
$$(x+5)^2+(y+5)^2<25,\ x\in[-10;0], y\in[-6,5;0]. \tag{9}$$

Optimal value: $f_{min}(-3,1302468;-1,5821422)=-106,7645367.$

The last classical function for testing algorithms and methods of swarm intelligence is the Simionescu function with functional limitations. It looks like this:

$$f(x,y)=0,1xy\to\min,$$
$$x^2+y^2<\left(1+0,2\cos\left(8\,arctg\left(x/y\right)\right)\right)^2,\ x\in[-1,25;1,25]; y\in[-1,25;1,25]. \tag{10}$$

Optimal value: $f_{min}(\pm0,84852813;\mp0,84852813)=-0,0720.$

3.4. Results of testing methods and algorithms and their modifications on ordinary functions

The software package for testing methods and algorithms was developed in the MATLAB 2023b computer mathematics system. Processor on which the calculations were carried out: INTEL Xeon E-2288G 8C/16T/3.7GHz/16MB/FCLGA1151/TRAY (CM8068404224102).

This section of the paper contains practical results that demonstrate the effectiveness of both classical methods and algorithms of global multidimensional optimization, as well as modifications obtained for them in the work. Results of comparison of the classical algorithm of global optimization by a swarm of particles and its modification. Table 1 shows the results of the computational algorithm for the hyperparameters $v^{min} = -5$, $v^{max} = 5$. Variable N Table 1 shows the number of elements in the swarm (population dimension), a_1, a_2 – the hyperparameters of the algorithm, which were described above in the pseudocode. The calculation error $\varepsilon = 10^{-12}$.

Table 1. Comparative Analysis of the Classical Algorithm of Global Multivariate Optimization by a Flock of Particles and Its Modifications

Function, Algorithm Parameters (Classic/Modified)	Space	Average number of iterations		Working hours of the program, Sec.		Deviation from the exact value of the objective function	
		Classical algorithm	Modified algorithm	Classical algorithm	Modified algorithm	Classical algorithm	Modified algorithm
Problem (6), N = 200; a_1=2, a_2 – rand(0; 4) / a_1=2, a_2 = 3	2	112	59	2,700E-07	9,998E-08	4,969E-13	3,040E-13
Problem (6), N = 600, a_1=2, a_2 – rand(0; 4) / a_1=2, a_2 = 3	5	48	29	1,880E-06	1,160E-06	8,319E-13	7,199E-13
Problem (6), N = 350, a_1=2, a_2 – rand(0; 4) / a_1=2, a_2 = 3	20	629	373	2,688E-05	1,590E-05	8,869E-13	8,949E-13
Problem (7), N = 350; a_1=2, a_2 – rand(0; 4) / a_1=2, a_2 = 2	2	85	63	2,800E-07	1,940E-07	7,199E-13	6,409E-13
Problem (8), N = 350; a_1=2, a_2 – rand(0; 4) / a_1=2, a_2 = 2	2	56	35	3,719E-07	1,260E-07	5,879E-13	4,859E-13
Problem (9), N = 350; a_1=2, a_2 – rand(0; 4) / a_1=2, a_2 = 2	2	212	69	6,803E-06	2,232E-06	5,659E-13	5,969E-13
Problem (10), N = 350; a_1=2, a_2 – rand(0; 4) / a_1=2, a_2 = 2	2	198	58	6,564E-06	2,544E-06	5,081E-13	5,001E-13

As can be seen from the results obtained, modifications of the algorithm give an advantage in the total number of iterations, which reduces the total load on the processor and reduces the total number of calculations to obtain the optimum of a function with a given accuracy (computational error that is the result of deviation from the exact value for test functions / functionals). Similarly, a study was conducted with a modification of the bee algorithm. Table 2 shows the results of the computational algorithm for the hyperparameters $Z_o = 150$, $Z_e = 100$, $\eta_{max} = 1$. The calculation error $\varepsilon = 10^{-12}$. The studies take into account the dependence (5), the value of the parameter $K = 5$.

Table 2. Comparative Analysis of the Classical Bee Algorithm of Global Multivariate Optimization and its Modifications

Function, Algorithm Parameters (Classic/Modified)	Space	Average number of iterations		Working hours of the program, Sec.		Deviation from the exact value of the objective function	
		Classical algorithm	Modified algorithm	Classical algorithm	Modified algorithm	Classical algorithm	Modified algorithm
Problem (6)	2	37	26	2,700E-07	1,998E-08	4,969E-13	3,039E-13
Problem (6)	5	138	84	1,880E-02	1,160E-02	8,319E-13	7,199E-13
Problem (6)	20	597	354	2,688E-05	1,590E-05	8,868E-13	8,948E-13
Problem (7)	2	48	33	2,800E-06	1,940E-06	7,199E-13	6,409E-13
Problem (8)	2	84	53	3,719E-07	1,260E-07	5,879E-13	4,859E-13
Problem (9)	2	182	121	6,803E-06	2,232E-06	5,659E-13	5,969E-13
Problem (10)	2	171	100	7,131E-06	2,541E-06	5,217E-13	5,021E-13

The results of modifying the classical algorithm give an acceleration of about 1.5–2 times compared to the classical algorithm. Table 3 presents the results of the study with a modification of the method of differential evolution. The table shows the results of the computational algorithm, taking into account (3) and (4). Calculation error ε = 10-12. The studies take into account the update of the value of the parameter F = 2 rand( ) every K iterations, with K = 5.

Table 3. Comparative Analysis of the Classical Method of Differential Evolution for Global Multivariate Optimization and Its Modification

Function, Method Parameters (Classic/Modified)	Space	Average number of iterations		Working hours of the program, Sec.		The resulting value of the function	
		Classical method	Modified Method	Classical method	Modified Method	Classical method	Modified Method
Problem (6)	2	8	5	1,500E-07	5,555E-08	2,761E-13	1,689E-13
Problem (6)	5	28	17	1,044E-06	6,443E-07	4,621E-13	3,999E-13
Problem (6)	20	371	220	1,493E-05	8,832E-06	4,927E-13	4,971E-13
Problem (7)	2	89	37	1,555E-07	1,078E-07	3,999E-13	3,561E-13
Problem (8)	2	64	41	2,066E-07	6,999E-08	3,266E-13	2,700E-13
Problem (9)	2	125	72	3,779E-06	1,240E-06	3,144E-13	3,316E-13
Problem (10)	2	97	43	3,251E-06	1,014E-06	3,544E-13	3,154E-13

3.5. Non-linear regression model

Let's consider the basic principles of finding a solution to a problem using a specific model mathematical example. Let the mathematical model that describes the process look like this:

$$y = \frac{b_1 + b_2 x + b_3 x^2 + b_4 x^3}{1 + b_5 x + b_6 x^2 + b_7 x^3}.$$

Task: to find such values of parameters $b_i, i = \overline{1;7}$, that the mathematical model describes statistical data as accurately as possible.

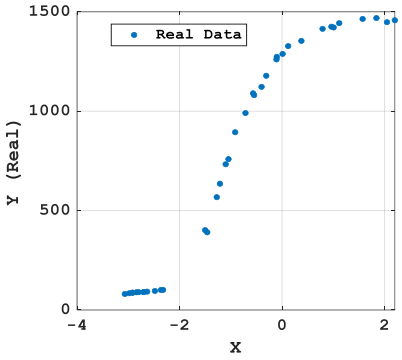
Loss Function is a target function that has one of the following forms:

$$E(b_1,\ldots,b_7) = \sqrt{\frac{1}{S}\sum_{i=1}^{S}\left(y_i - y_i^{Model}(b_1,\ldots,b_7)\right)^2}; \quad E(b_1,\ldots,b_7) = \frac{1}{S}\sum_{i=1}^{S}\left|y_i - y_i^{Model}(b_1,\ldots,b_7)\right|;$$

$$E(b_1,\ldots,b_7)=\max_i\left|y_i-y_i^{Model}(b_1,\ldots,b_7)\right|,$$

y_i^{Model} is the value of the functional dependence for the set $(b_1,\ldots,b_7)$ in the *i*-th observation, *S* is the number of selected observations. Dataset: 36 observations – 1 independent variable (*x*), 1 dependent variable (*y*). The data looks like Table 4:

Table 4. Input data for building a nonlinear regression model and their visualization

y	x	y	x	y	x	Graphical representation of data
80,574	-3,067	401,672	-1,501	1273,514	-0,103	
84,248	-2,981	390,724	-1,460	1288,339	0,010	
87,264	-2,921	567,534	-1,274	1327,543	0,119	
87,195	-2,912	635,316	-1,212	1353,863	0,377	
89,076	-2,840	733,054	-1,100	1414,509	0,790	
89,608	-2,797	759,087	-1,046	1425,208	0,963	
89,868	-2,702	894,206	-0,915	1421,384	1,006	
90,101	-2,699	990,785	-0,714	1442,962	1,115	
92,405	-2,633	1090,109	-0,566	1464,350	1,572	
95,854	-2,481	1080,914	-0,545	1468,705	1,841	
100,696	-2,363	1122,643	-0,400	1447,894	2,047	
101,060	-2,322	1178,351	-0,309	1457,628	2,200	

The search for a solution to the problem of finding unknown coefficients of the nonlinear regression model was carried out by steps in the general methodology of machine learning.

1) Shuffle the sample with the rows in the database that contains the observations (database rows are swapped randomly).
2) Selection from the beginning of most of the data for training (search for parameters $b_1^*,\ldots,b_7^*$, that best describe the statistics). This is an educational (training sample). Typically, this subsample contains 75 % to 80 % of all observations in the sample.
3) Finding $b_1^*,\ldots,b_7^*$ parameters using the optimization method / algorithm of the objective error function.
4) Finding an error on the test training and test subsamples for the found $b_1^*,\ldots,b_7^*$. ones They are compared with each other, conclusions are formulated.
5) For the found, $b_1^*,\ldots,b_7^*$ dependency graphs are constructed: real data on the test sample and model data on the training sample; real data on the test sample and model data on the test sample.
6) The results of the dependence of real data on the test sample and model data on the training sample and real data on the test sample and model data on the test sample are compared with the dependence $y = x$.
7) Graphical derivation of the statistical dependence of the independent variable (*x*) and the dependent variable (*y*) on the entire sample. Imposition of functional dependence on a given coordinate system

$$y=\frac{b_1^*+b_2^*x+b_3^*x^2+b_4^*x^3}{1+b_5^*x+b_6^*x^2+b_7^*x^3}.$$

If a functional dependency does describe a certain statistical process correctly, then the *Loss Function* will have relatively small values. The main results of the simulation are shown in Figure 2 and Figure 3.

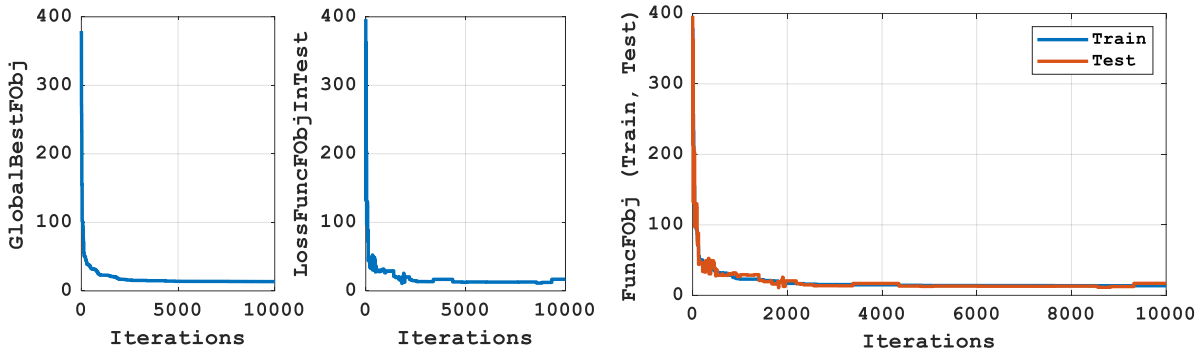


Fig. 2. Convergence process when searching for nonlinear regression parameters for test and training samples

It took more iterations to solve the nonlinear regression problem than to find the global extremum of the classical test functions, which are given above together with the results of the application of optimization methods and algorithms. It should also be noted that the adaptation of algorithms and methods of swarm intelligence to the tasks of nonlinear regression analysis can be an interesting area of research.

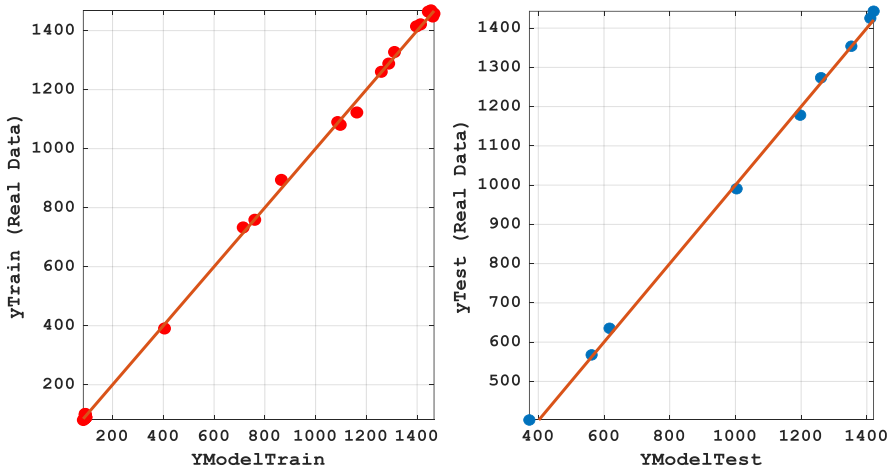


Fig. 3. Forecast graphs and real data for training and test samples

Most technical problems involve building optimization models on statistical data, which are mostly reduced to the form where the least squares method can be effectively applied. Even if the least-squares method is difficult to apply due to the poor conditionality of matrix structures, here regularization is performed and again algorithms and methods of swarm intelligence are used to minimize the generalized functionality. For most tasks of regression analysis, the results of machine learning are built in the space "Model data – real data". This result is shown in Figure 3, which shows the results of the "forecast" on the training sample (left), as well as the results of the "forecast" on the test sample (on the right).

3.6. Applied problems with functional limitations

In this section of the work, several applied problems are considered, which involve the use of swarm intelligence methods. Other well-known methods and algorithms of global multivariate optimization have worse results than those considered on similar applied problems.

3.6.1. The problem of minimizing the weight of the reducer [10]

The problem minimizes the weight of the gearbox (Figure 4) taking into account the constraints on wheel tooth bending, surface stress, transverse deflections of the shafts and stress in the shafts.

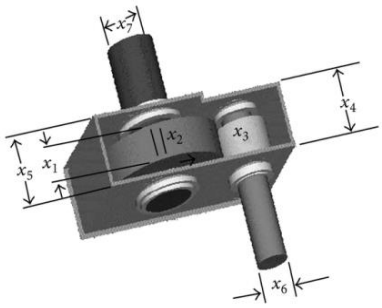


Fig. 4. Gearbox geometers

Problem parameters:

b – Plating width (x_1);

m – Wheel Module (x_2);

z – Number of wheel teeth (x_3 – integer variable);

l_1 – The length of the first shaft between the bearings (x_4);

l_2 – The length of the second shaft between the bearings (x_5);

d_1 – The diameter of the first shaft (x_6);

d_2 – The diameter of the second shaft (x_7).

The objective function looks like this:

$$f(\vec{x}) = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) -$$
$$-1.508x_1\left(x_6^2 + x_7^2\right) + 7.4777\left(x_6^3 + x_7^3\right) + 0.7854\left(x_4x_6^2 + x_5x_7^2\right) \to min,$$

$$\begin{cases} g_1(\vec{x}) = \dfrac{27}{x_1x_2^2x_3} - 1 \le 0,\ g_2(\vec{x}) = \dfrac{397.5}{x_1x_2^2x_3^2} - 1 \le 0,\ g_3(\vec{x}) = \dfrac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0, \\ g_4(\vec{x}) = \dfrac{1.93}{x_2x_3x_7^4} - 1 \le 0,\ g_5(\vec{x}) = \dfrac{1.0}{110x_6^3}\sqrt{\left(\dfrac{745.0x_4}{x_2x_3}\right)^2 + 16.9\times10^6} - 1 \le 0, \\ g_6(\vec{x}) = \dfrac{1.0}{85x_7^3}\sqrt{\left(\dfrac{745.0x_5}{x_2x_3}\right)^2 + 157.5\times10^6} - 1 \le 0,\ g_7(\vec{x}) = \dfrac{x_2x_3}{40} - 1 \le 0, \\ g_8(\vec{x}) = \dfrac{5x_2}{x_1} - 1 \le 0,\ \ g_9(\vec{x}) = \dfrac{x_1}{12x_2} - 1 \le 0,\ \ g_{10}(\vec{x}) = \dfrac{1.5x_6 + 1.9}{x_4} - 1 \le 0,\ \ g_{11}(\vec{x}) = \dfrac{1.1x_7 + 1.9}{x_5} - 1 \le 0, \\ 2,6 \le x_1 \le 3,6;\ 0,7 \le x_2 \le 0,8;\ 17 \le x_3 \le 28;\ 7,3 \le x_4 \le 8,3;\ x_3 \in \mathbb{Z}, \\ 7,8 \le x_5 \le 8,3; 2,9 \le x_6 \le 3,9; 5,0 \le x_7 \le 5,5. \end{cases} \quad (11)$$

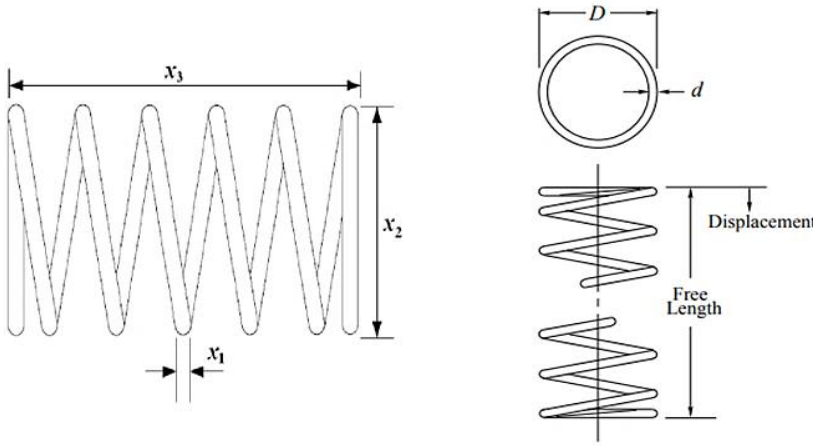
Table 5 shows the results of the method of differential evolution and its modification, which are described in the paper for 40 different runs of the program for searching for the global extremum. To generate the results in Table 5, the best of the three tools for finding the global extremum of test multivariate functions was chosen. The best tool is the method of differential evolution. Table 5 takes into account the corresponding modifications of the method that were used to generate the results in the previous tables.

Table 5. Computational Experiments for the Problem (11)

Computational Experiment Number	Number of iterations		Working hours of the program, Sec.		Deviations from the exact value of the objective function [10]	
	Classical method	Modified Method	Classical method	Modified Method	Classical method	Modified Method
1	356	211	1,28070E+00	7,59531E-01	9,89873E-13	4,26860E-13
2	366	193	1,35873E-02	8,11026E-03	2,50646E-13	3,07247E-13
3	390	167	1,85683E-02	8,61480E-03	2,60083E-13	2,34069E-13
…	…	…	…	…	…	…
38	464	140	9,22273E-04	7,14465E-04	2,31168E-13	1,31756E-13
39	444	159	2,23053E-04	5,71327E-04	7,10985E-13	6,39946E-13
40	432	138	3,47420E-04	2,24475E-04	4,80957E-13	6,19187E-13

3.6.2. The problem of optimizing the spring design [10]

The problem minimizes the tension / compression of the spring taking into account the limitations of minimum deviation, shear stress, overvoltage frequency, diameter constraints and design variables (Figure 5).



Parameters of the problem:
d – wire diameter (x_1),
D – average coil diameter (x_2),
N – number of active coils (x_3).

Fig. 5. Structural diagram of the spring and its parameters

The mathematical model of the optimization applied problem is as follows:

$$f(\vec{x}) = (x_3 + 2)\, x_2 x_1^2 \to min.$$

$$\begin{cases} g_1(\vec{x}) = 1 - \dfrac{x_2^3 x_3}{7{,}178 x_1^4} \le 0,\ g_3(\vec{x}) = 1 - \dfrac{140{,}45 x_1}{x_2^2 x_3} \le 0,\ g_4(\vec{x}) = \dfrac{x_2 + x_1}{1.5} - 1 \le 0, \\ g_2(\vec{x}) = \dfrac{4x_2^2 - x_1 x_2}{12{,}566\left(x_2 x_1^3\right) - x_1^4} + \dfrac{1}{5{,}108 x_1^2} - 1 \le 0, \\ 0{,}005 \le x_1 \le 2{,}0,\ \ 0{,}25 \le x_2 \le 1{,}3,\ \ 2{,}0 \le x_3 \le 15{,}0. \end{cases} \tag{12}$$

Table 5 shows the results of the method of differential evolution and its modification, which are described in the paper for 30 different runs of the program for searching for the global extremum.

Table 6. Computational Experiments for the Problem (12)

Computational Experiment Number	Number of iterations		Working hours of the program, Sec.		Deviations from the exact value of the objective function [10]	
	Classical algorithm	Modified algorithm	Classical algorithm	Modified algorithm	Classical algorithm	Modified algorithm
1	356	211	9,32504E-01	4,08713E-01	9,13226E-13	2,54923E-13
2	386	203	8,68428E-01	5,38674E-01	7,05207E-13	1,71713E-13
3	400	224	2,17573E-01	1,22818E-01	3,92392E-13	1,35203E-13
…	…	…	…	…	…	…
28	371	196	5,64807E-01	3,94011E-01	1,66983E-13	1,83788E-13
29	395	213	6,04579E-01	4,29167E-01	1,45964E-13	2,54636E-13
30	402	231	2,17123E-01	1,32962E-01	3,66866E-13	1,89860E-13

As we can see from the results of testing the method, it has proven to be very effective on applied problems. A modification of the classical method showed an acceleration of about 1.5–2 times.

4. Discussion

The main advantage of modern methods and algorithms of swarm intelligence is the intuitive structure of the natural origin of the swarm, which means that it is a relatively simple software implementation. Therefore, it makes swarm intelligence tools popular in various fields of research.

Most processes are structurally nonlinear. Swarm intelligence demonstrates high efficiency in finding global optima for a wide range of functions, including nonlinear and non-convex functions, and works effectively with nonlinear functions through the appropriate mechanism of evolution, which allows you to move in the parameter space to find the global optimum. Another of the main advantages of the methods and algorithms under consideration is the small number of hyperparameters that need to be configured. This simplifies the use of methods and algorithms and customization compared to other algorithms.

5. Conclusions

The development of computing technologies makes it possible to simulate the behavior of complex objects and systems. The development of new and modification of existing systems involves the construction of mathematical models in the form of optimization in order to study various parameters of such systems. Such models make it possible to have an idea of the modes of operation of various objects and systems, their optimal parameters (settings), geometric properties of such systems. The use of swarm intelligence contributes to the study and development of such objects and systems, since the usual approximate methods of global multivariate optimization do not have such an opportunity due to the complexity and nonlinearity of functional dependencies that describe systems and their operation.

In this work, three relevant methods and algorithms of swarm intelligence are studied: the algorithm of global optimization by a swarm of particles, the bee algorithm for finding global solutions to problems that are presented in extreme statements, and the method of differential evolution. These methods have been tested on various scientific and applied problems, including problems that boil down to the search for global extremes of multivariate functions with and without constraints, as well as on problems that reduce to the use of nonlinear multivariate regression models and forecasting time series that arise during the study of the work of various system complexes. Three modifications of the considered methods and algorithms have been obtained. The first modification consists in replacing the worst element of the population with the average position of the element throughout the population. This approach works effectively when applied at each iteration or periodically at every K iteration. The second modification of these methods and algorithms is to apply the principle of deterministic movement towards the best element of the population. Movement step h it is determined in proportion (according to the linear law) to the circumference of the population, which will coincide to the global optimum over time. The third modification consists in changing the hyperparameters of optimization methods/algorithms. This involves changing one or more hyperparameters in the course of the program's operation. The practical results of the article are that the complex application of three modifications for each method/algorithm gives advantages in increasing the dimension of search in an extreme problem. For multivariate problems, modifications affect the elements of the population and allow for a more accurate definition of the solution. This reduces the total number of iterations of the method/algorithm, which means that it reduces the time to find the optimum of the problem with a given accuracy.

References

1. Khaidurov, V., Tatenko, V., Lytovchenko, M., Tsiupii, T., & Zhovnovach, T. (2024). Methods and Algorithms of Swarm Intelligence for the Problems of Nonlinear Regression Analysis and Optimization of Complex Processes, Objects, and Systems: Review and Modification of Methods and Algorithms. *System Research in Energy*, *3*(79), 46−61. https://doi.org/10.15407/srenergy2024.03.046
2. Ewees, A. A., Gaheen, M. A., Yaseen, Z. M., & Ghoniem, R. M. (2022). Grasshopper Optimization Algorithm with Crossover Operators for Feature Selection and Solving Engineering Problems. *IEEE Access*, 10, 23304–23320. https://doi.org/10.1109/ACCESS.2022.3153038
3. Yu, Y., Liu, J., & Wei, C. (2022). Hawk and pigeon's intelligence for UAV swarm dynamic combat game via competitive learning pigeon-inspired optimization. *Science China Technological Sciences*, *65*(5), 1072–1086. https://doi.org/10.1007/s11431-021-1951-9
4. Pradhan, C., Senapati, M. K., Ntiakoh, N. K., & Calay, R. K. (2022). Roach Infestation Optimization MPPT Algorithm for Solar Photovoltaic System. *Electronics*, *11*(6), 927. https://doi.org/10.3390/electronics11060927
5. Hassanien, A. E., & Emary, E. (2016). *Swarm intelligence: principles, advances, and applications*. CRC Press is an imprint of Taylor & Francis Group, an Informa business. ISBN: 978-1-4987-4107-1.
6. Manjarres, A. V., Sandoval, L. G. M., & Suarez, M. J. S. (2018). Data Mining Techniques Applied in Educational Environments: Literature Review. *Digital Education Review*, 33, 235–266. https://doi.org/10.1344/der.2018.33.235-266
7. Prabha, S. L., Dr. Shanavas, A. R. M. (2015). Application of Educational Data mining techniques in e-Learning A Case Study. *International Journal of Computer Science and Information Technologies*, *6*(5), 4440–4443. Retrieved from https://www.ijcsit.com/docs/Volume%206/vol6issue05/ijcsit2015060562.pdf
8. Yılmaz, S., & Kucuksille, E. U. (2015). A New Modification Approach on Bat Algorithm for Solving Optimization Problems. *Applied Soft Computing*, 28, 259−275. https://doi.org/10.1016/j.asoc.2014.11.029

9. Wong, L.-H., & Looi, C.-K. (2009). Adaptable Learning Pathway Generation with Ant Colony Optimization. *Educational Technology & Society*, *12*(3), 309–326. Retrieved November 11, 2024, from https://www.j-ets.net/collection/published-issues/12_3
10. Rao, S. S. (2009). *Engineering Optimization Theory and Practice*. Fourth Edition. ISBN 978-0-470-18352-6.

МОДИФІКАЦІЯ МЕТОДІВ РОЙОВОГО ІНТЕЛЕКТУ ДЛЯ ЗАДАЧ ОПТИМІЗАЦІЇ СКЛАДНИХ ПРОЦЕСІВ, ОБ'ЄКТІВ І СИСТЕМ

Владислав Хайдуров[1,2]*, канд. техн. наук, ст. досл., https://orcid.org/0000-0002-4805-8880
Артем Зінченко[1], канд. техн. наук, доцент, https://orcid.org/0000-0003-1586-3645
Тамара Цюпій[3], канд. фіз.-мат. наук, доцент, https://orcid.org/0000-0003-2206-2897
Роман Яровий[4], канд. техн. наук, https://orcid.org/0000-0001-8978-8137
[1]Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Берестейський просп., 37, Київ, 03056, Україна;
[2]Інститут загальної енергетики НАН України, вул. Антоновича, 172, Київ, 03150, Україна;
[3]Національний університет біоресурсів і природокористування України, вул. Героїв Оборони, 15, Київ, 03041, Україна;
[4]ПЗВО «Європейський університет», бульвар Академіка Вернадського, 16 В, Київ, 03041, Україна
*Автор-кореспондент: allif0111@gmail.com

Анотація. *Розробка швидкісних методів й алгоритмів глобальної багатовимірної оптимізації і їх модифікацій у різних сферах науки, техніки, економіки є актуальним завданням, яке передбачає зменшення обчислювальних затрат, прискорення і ефективний пошук розв'язків такого роду задач. У зв'язку з тим, що більшість серйозних завдань передбачають пошук десятків, сотень або тисяч оптимальних параметрів математичних моделей, простір пошуку цих параметрів зростає нелінійно. Сучасний ройовий інтелект має значний потенціал для застосування в енергетичній галузі через свою здатність до оптимізації та розв'язання складних проблем. За допомогою нього можна вирішувати науково-прикладні задачі оптимізації споживання енергії в будівлях, промислових комплексах та міських системах, зменшуючи втрати енергії та підвищуючи ефективність використання ресурсів, а також для побудови різних елементів енергетичних систем загалом. Відомі методи й алгоритми ройового інтелекту також активно застосовують для прогнозування виробництва енергії від відновлюваних джерел, таких як сонячна та вітрова енергія. Це дозволяє краще управляти джерелами енергії та планувати їхнє використання. Актуальність модифікацій методів й алгоритмів зумовлена питаннями прискорення швидкості їх роботи під час розв'язання задач машинного навчання, зокрема у моделях нелінійної регресії, задачах класифікації, де кількість спостережуваних даних може сягати десяти і сотні тисяч або більше. У роботі розглянуті й модифіковані відомі ефективні методи й алгоритми ройового інтелекту (алгоритм оптимізації роєм частинок, бджолиний алгоритм оптимізації, метод диференціальної еволюції) для пошуку розв'язків багатовимірних екстремальних задач з обмеженнями і без обмежень, а також задач нелінійного регресійного аналізу. Продемонстрована ефективність роботи модифікованих методів на різних класичних і прикладних задачах, які використовуються в проєктуванні елементів складних об'єктів та їх систем. Проведено порівняльний аналіз результатів роботи даних методів.*
Ключові слова: оптимізація, ройовий інтелект, математичне моделювання, нелінійна регресія, складні об'єкти та системи.

Надійшла до редколегії: 27.01.2025