ISSN 2786-7102 (Online), ISSN 2786-7633 (Print)
https://doi.org/10.15407/srenergy2024.03.046

UDC 517.9:519.6

Vladyslav Khaidurov[1,2]*, PhD (Engin.), Senior Researcher, https://orcid.org/0000-0002-4805-8880
Vadym Tatenko[1], https://orcid.org/0009-0008-4869-9689
Mykyta Lytovchenko[1], https://orcid.org/0009-0001-6671-7763
Tamara Tsiupii[3], PhD (Engin.), Associate Professor, https://orcid.org/0000-0003-2206-2897
Tetiana Zhovnovach[4], https://orcid.org/0000-0003-1037-4383
[1]National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37, Beresteiskyi Avenue., 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]Cherkasy branch of European University, 83, Smilyanska St., Cherkasy, 18008, Ukraine
*Corresponding author: allif0111@gmail.com

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

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 problem that involves reducing computing costs, accelerating, and effectively searching for solutions to such problems. Since 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. Currently, there are many modern methods and algorithms of swarm intelligence that solve today's scientific and applied problems, but they require modifications due to the large spaces of searching for optimal model parameters. Modern swarm intelligence has significant potential for application in the energy industry due to its ability to optimize and solve complex problems. It can be used 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 applied 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, and 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 obtained modifications of the well-known classic effective methods and algorithms of swarm intelligence, which are present in the work, effectively solve complex scientific and applied tasks of designing complex objects and systems. A comparative analysis of methods and algorithms will be conducted in the next study on this topic.*

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 [4–6].

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 [7, 8]. 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 [9, 10]. 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 [11–13]. 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 [14–16].

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 [17].

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 [18, 19]:

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

where $\mathbf{X} = \left(X_1, X_2, \cdots, X_n\right)$, 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(**X**) is often subject to functional constraints, which are presented in the form of the following functional dependencies:

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

The complexity of the objective function F(**X**) and the corresponding constraints g_i(**X**) is determined by a specific technical problems [20; 21]. The appearance *of* F(**X**) and g_i(**X**) determines the choice of the optimization method that will be used to search for $\mathbf{X}^* = \left(X_1^*, X_2^*, \cdots, X_n^*\right)$, $F(\mathbf{X}^*)$.

It should be noted that the dependence of the species:

$$y = F\left(\mathbf{X}\right) \to \min,$$

In machine learning problems, it is often presented as a summary (or average) error, which consists of a deviation between real data and model data, which is skipped due to functional dependence. 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.

To begin with, let's consider a one-factor (factor (independent) variable – *x*, dependent variable – *y*) regression model. Let us assume that we know the general form of a mathematical model that describes a certain process. It looks like $y = F\left(\beta_j, x\right), \ j = \overline{1;N}.$ In the problem of regression analysis, it is assumed that there are statistical data that are presented in the form of observations: $\left(x_i, y_i\right), \ i = \overline{1;M}, \ M \ge N.$ Knowing the appearance $y = F\left(\beta_j, x\right), \ j = \overline{1;N}$ of the model and observational data $\left(x_i, y_i\right), \ i = \overline{1;M}, \ M \ge N,$ it is necessary to find the parameters $\beta_j^*, \ j = \overline{1;N},$ under which the mathematical model is formed $y = F\left(\beta_j^*, x\right)$ It most accurately describes a certain process that takes place in a particular system.

That is, for such a problem, we can write a system of equations in the form:

$$\begin{gathered} y_1 = F\left(\beta_1, \beta_2, \cdots, \beta_N, x_1\right), y_2 = F\left(\beta_1, \beta_2, \cdots, \beta_N, x_2\right), \\ y_3 = F\left(\beta_1, \beta_2, \cdots, \beta_N, x_3\right), \cdots, \ y_M = F\left(\beta_1, \beta_2, \cdots, \beta_N, x_M\right). \end{gathered} \tag{1}$$

But, obviously, the statistics are obtained with some error. This means that in reality we do not have a system of form (1), but a system of the form:

$$\begin{gathered} y_1 = F\left(\beta_1, \beta_2, \cdots, \beta_N, x_1\right) + \varepsilon_1, y_2 = F\left(\beta_1, \beta_2, \cdots, \beta_N, x_2\right) + \varepsilon_2, \\ y_3 = F\left(\beta_1, \beta_2, \cdots, \beta_N, x_3\right) + \varepsilon_3, \cdots, y_M = F\left(\beta_1, \beta_2, \cdots, \beta_N, x_M\right) + \varepsilon_M, \end{gathered} \tag{2}$$

where $\varepsilon_i, \ i = \overline{1;M}$ is the measurement error on the *i*-th observation. In this case, for (2) it is necessary to find such values of the parameters in order to minimize the total error that was obtained as a result of measurements. To do this, the well-known least-squares method is used: $\beta_j^*, \ j = \overline{1;N},$

$$\begin{gathered} E = \left|\varepsilon_1\right| + \left|\varepsilon_2\right| + \left|\varepsilon_3\right| + \cdots + \left|\varepsilon_M\right| = \sum_{j=1}^{M} \left|\varepsilon_j\right| \to \min, \\ \left(E = \frac{1}{M}\left(\left|\varepsilon_1\right| + \left|\varepsilon_2\right| + \left|\varepsilon_3\right| + \cdots + \left|\varepsilon_M\right|\right) = \frac{1}{M}\sum_{j=1}^{M} \left|\varepsilon_j\right| \to \min \right) \end{gathered} \tag{3}$$

or

$$\begin{gathered} E = \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \cdots + \varepsilon_M^2 = \sum_{j=1}^{M} \varepsilon_j^2 \to \min, \\ \left(E = \frac{1}{M}\left(\varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \cdots + \varepsilon_M^2\right) = \frac{1}{M}\sum_{j=1}^{M} \varepsilon_j^2 \to \min \right), \end{gathered} \tag{4}$$

or

$$E = \max_j \left|\varepsilon_j\right| \to \min. \tag{5}$$

For gradient methods, the first expression (3) is used for E. For swarm intelligence methods, it does not matter which of the formulas (3), (4) or (5) is used as a criterion for the minimum error. It should be noted that option (3) for E (through the sum of the modules) is less commonly used in gradient methods, since the derivative of $y(t)=|t|$ the function at the point of the extremum does not exist. In this case, we have a value E based on formula (3):

$$\begin{gathered} E=\left(y_1-F\left(\beta_1,\beta_2,\cdots,\beta_N,x_1\right)\right)^2+ \\ +\left(y_2-F\left(\beta_1,\beta_2,\cdots,\beta_N,x_2\right)\right)^2+\ldots+\left(y_M-F\left(\beta_1,\beta_2,\cdots,\beta_N,x_M\right)\right)^2. \end{gathered} \tag{6}$$

Let's take the classic approach. Let us find the gradient vector for (6) and equate it to zero (a necessary condition for the extremum of the function / functional):

$$\begin{aligned} E'_{\beta_1}&=-2\left(y_1-F\left(\beta_1,\beta_2,\cdots,\beta_N,x_1\right)\right)F'_{\beta_1}\left(\beta_1,\beta_2,\cdots,\beta_N,x_1\right)- \\ &-2\left(y_2-F\left(\beta_1,\beta_2,\cdots,\beta_N,x_2\right)\right)F'_{\beta_1}\left(\beta_1,\beta_2,\cdots,\beta_N,x_2\right)-\cdots- \\ &-2\left(y_M-F\left(\beta_1,\beta_2,\cdots,\beta_N,x_M\right)\right)F'_{\beta_1}\left(\beta_1,\beta_2,\cdots,\beta_N,x_M\right)=0. \\ &\cdots \\ E'_{\beta_N}&=-2\left(y_1-F\left(\beta_1,\beta_2,\cdots,\beta_N,x_1\right)\right)F'_{\beta_N}\left(\beta_1,\beta_2,\cdots,\beta_N,x_1\right)- \\ &-2\left(y_2-F\left(\beta_1,\beta_2,\cdots,\beta_N,x_2\right)\right)F'_{\beta_N}\left(\beta_1,\beta_2,\cdots,\beta_N,x_2\right)-\cdots- \\ &-2\left(y_M-F\left(\beta_1,\beta_2,\cdots,\beta_N,x_M\right)\right)F'_{\beta_N}\left(\beta_1,\beta_2,\cdots,\beta_N,x_M\right)=0. \end{aligned} \tag{7}$$

From dependencies (7) we obtain a system of nonlinear (in the general case) equations of the form:

$$\begin{cases} \sum\limits_{i=1}^{M}\left[F'_{\beta_1}\left(\beta_1,\beta_2,\cdots,\beta_N,x_i\right)\cdot\left(F\left(\beta_1,\beta_2,\cdots,\beta_N,x_i\right)-y_i\right)\right]=0, \\ \sum\limits_{i=1}^{M}\left[F'_{\beta_2}\left(\beta_1,\beta_2,\cdots,\beta_N,x_i\right)\cdot\left(F\left(\beta_1,\beta_2,\cdots,\beta_N,x_i\right)-y_i\right)\right]=0, \\ \cdots \\ \sum\limits_{i=1}^{M}\left[F'_{\beta_N}\left(\beta_1,\beta_2,\cdots,\beta_N,x_i\right)\cdot\left(F\left(\beta_1,\beta_2,\cdots,\beta_N,x_i\right)-y_i\right)\right]=0. \end{cases} \tag{8}$$

Obviously, system (8) has N equations and N unknowns. Iterative methods and algorithms are often applied to this kind of system, for example, the Newtonian method, the Levenberg-Marquardt method, quasi-Newtonian methods, etc.

For example, Newton's method for a system of the form:

$$\begin{cases} f_1\left(\beta_1,\beta_2,\cdots,\beta_N\right)=0, \\ f_2\left(\beta_1,\beta_2,\cdots,\beta_N\right)=0, \\ \cdots \\ f_N\left(\beta_1,\beta_2,\cdots,\beta_N\right)=0 \end{cases} \tag{9}$$

will look like this:

$$\begin{pmatrix} \beta_1^{(k+1)} \\ \beta_2^{(k+1)} \\ \cdots \\ \beta_N^{(k+1)} \end{pmatrix} = \begin{pmatrix} \beta_1^{(k)} \\ \beta_2^{(k)} \\ \cdots \\ \beta_N^{(k)} \end{pmatrix} - J^{-1}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right) \begin{pmatrix} f_1\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right) \\ f_2\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right) \\ \cdots \\ f_N\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right) \end{pmatrix}, \tag{10}$$

where J is the Jacobian at the point $\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right)$, which is of the form:

$$J\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right) =$$

$$= \begin{pmatrix} \frac{\partial f_1}{\partial \beta_1}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right) & \cdots & \frac{\partial f_1}{\partial \beta_N}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial \beta_1}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right) & \cdots & \frac{\partial f_N}{\partial \beta_N}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right) \end{pmatrix}, \tag{11}$$

where k is the iteration number. To start such an iterative process, (9)–(11) specify an initial approximation $\left(\beta_1^{(0)},\beta_2^{(0)},\cdots,\beta_N^{(0)}\right)$. The stopping criterion can be considered the fulfillment of one of the following conditions:

$$\left|\overline{\beta^{(k+1)}} - \overline{\beta^{(k)}}\right| = \sqrt{\sum_{i=1}^{N}\left(\beta_i^{(k+1)} - \beta_i^{(k)}\right)^2} < \varepsilon, \varepsilon > 0, \tag{12}$$

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(f_i\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right)\right)^2} < \varepsilon, \varepsilon > 0, \tag{13}$$

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left|f_i\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right)\right|} < \varepsilon, \varepsilon > 0, \tag{14}$$

$$\max_i \left|f_i\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right)\right| < \varepsilon, \varepsilon > 0. \tag{15}$$

Several conditions for stopping an iterative process can be applied, and not only (12), (13), (14), and (15).

Taking into account the above, it is possible to rewrite the iterative formula of Newton's classical method (10), (11) for a system (8) of nonlinear (in the general case) equations:

$$\begin{pmatrix} \beta_1^{(k+1)} \\ \beta_2^{(k+1)} \\ \cdots \\ \beta_N^{(k+1)} \end{pmatrix} = \begin{pmatrix} \beta_1^{(k)} \\ \beta_2^{(k)} \\ \cdots \\ \beta_N^{(k)} \end{pmatrix} - J^{-1}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right) \times$$

$$\times \begin{pmatrix} \sum_{i=1}^{M}\left[F'_{\beta_1}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\cdot\left(F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)-y_i\right)\right] \\ \sum_{i=1}^{M}\left[F'_{\beta_2}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\cdot\left(F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)-y_i\right)\right] \\ \cdots \\ \sum_{i=1}^{M}\left[F'_{\beta_N}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\cdot\left(F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)-y_i\right)\right] \end{pmatrix}, \tag{16}$$

where J is the Jacobian at the point $\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right)$, which is of the form:

$$J\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right)=\sum_{i=1}^{M}\left[\begin{array}{c}\frac{\partial^2 F}{\partial\beta_k\partial\beta_l}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\times\left(F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)-y_i\right)+\\ +\frac{\partial F}{\partial\beta_k}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\times\frac{\partial F}{\partial\beta_l}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\end{array}\right], \quad (17)$$

where k is the iteration number. To start such an iterative process, (16)–(17) specify an initial approximation $\left(\beta_1^{(0)},\beta_2^{(0)},\cdots,\beta_N^{(0)}\right)$. The stopping criterion can be considered the fulfillment of one of the following conditions:

$$\left|\overline{\beta^{(k+1)}}-\overline{\beta^{(k)}}\right|=\sqrt{\sum_{i=1}^{N}\left(\beta_i^{(k+1)}-\beta_i^{(k)}\right)^2}<\varepsilon,\varepsilon>0, \quad (18)$$

$$\sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(\sum_{i=1}^{M}\left[\begin{array}{l}F'_{\beta_j}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\times\\ \times\left(F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)-y_i\right)\end{array}\right]\right)^2}<\varepsilon,\varepsilon>0, \quad (19)$$

$$\sqrt{\frac{1}{N}\sum_{j=1}^{N}\left|\sum_{i=1}^{M}\left[\begin{array}{l}F'_{\beta_j}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\times\\ \times\left(F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)-y_i\right)\end{array}\right]\right|}<\varepsilon,\varepsilon>0, \quad (20)$$

$$\max_j\left|\sum_{i=1}^{M}\left[\begin{array}{l}F'_{\beta_j}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\times\\ \times\left(F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)-y_i\right)\end{array}\right]\right|<\varepsilon,\varepsilon>0. \quad (21)$$

Here, too, any of the conditions (18), (19), (20) and (21) can be used to stop the iterative process of finding the optimal parameters of the nonlinear regression model.

To simplify the calculations of the first derivatives in (16), (17), numerical approximations are used, which are derived on the basis of Taylor series, for example:

$$F'_{\beta_j}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\approx\frac{1}{\Delta\beta_j}\left[\begin{array}{l}F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_j^{(k)}+\Delta\beta_j,\cdots,\beta_N^{(k)},x_i\right)-\\ -F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_j^{(k)},\cdots,\beta_N^{(k)},x_i\right)\end{array}\right], \quad (22)$$

or

$$F'_{\beta_j}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\approx\frac{1}{\Delta\beta_j}\left[\begin{array}{l}F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_j^{(k)},\cdots,\beta_N^{(k)},x_i\right)-\\ -F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_j^{(k)}-\Delta\beta_j,\cdots,\beta_N^{(k)},x_i\right)\end{array}\right], \quad (23)$$

or

$$F'_{\beta_j}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\approx\frac{1}{2\Delta\beta_j}\left[\begin{array}{l}F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_j^{(k)}+\Delta\beta_j,\cdots,\beta_N^{(k)},x_i\right)-\\ -F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_j^{(k)}-\Delta\beta_j,\cdots,\beta_N^{(k)},x_i\right)\end{array}\right], \quad (24)$$

or

$$F'_{\beta_j}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\approx\frac{1}{\Delta\beta_j}\left[\begin{array}{l}F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_j^{(k)}+\dfrac{\Delta\beta_j}{2},\cdots,\beta_N^{(k)},x_i\right)-\\ -F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_j^{(k)}-\dfrac{\Delta\beta_j}{2},\cdots,\beta_N^{(k)},x_i\right)\end{array}\right]. \quad (25)$$

It should be noted that each of the formulas (22), (23), (24) and (25) has its own error, which is obtained from the decomposition of the function/functional into the Taylor series according to the corresponding parameters.

To simplify the calculations of the second derivatives in (16) and (17), numerical approximations are also used, which are derived from the Taylor series:

$$\frac{\partial^2 F}{\partial\beta_j^2}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)\approx\frac{1}{\left(\Delta\beta_j\right)^2}\times\left[\begin{array}{c}F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_j^{(k)}+\Delta\beta_j,\cdots,\beta_N^{(k)},x_i\right)-\\ -2F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_j^{(k)},\cdots,\beta_N^{(k)},x_i\right)+\\ +F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_j^{(k)}-\Delta\beta_j,\cdots,\beta_N^{(k)},x_i\right)\end{array}\right], \quad (26)$$

$$\frac{\partial^2 F}{\partial\beta_k\beta_l}\left(\beta_1^{(k)},\cdots,\beta_N^{(k)},x_i\right)\approx\frac{1}{4\Delta\beta_k\Delta\beta_l}\times\left[\begin{array}{c}F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_k^{(k)}+\Delta\beta_k,\cdots,\beta_l^{(k)}+\Delta\beta_l,\cdots,\beta_N^{(k)},x_i\right)+\\ +F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_k^{(k)}-\Delta\beta_k,\cdots,\beta_l^{(k)}-\Delta\beta_l,\cdots,\beta_N^{(k)},x_i\right)-\\ -F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_k^{(k)}+\Delta\beta_k,\cdots,\beta_l^{(k)}-\Delta\beta_l,\cdots,\beta_N^{(k)},x_i\right)-\\ -F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_k^{(k)}-\Delta\beta_k,\cdots,\beta_l^{(k)}+\Delta\beta_l,\cdots,\beta_N^{(k)},x_i\right)\end{array}\right], \quad (27)$$

$$k=\overline{1;N},\ l=\overline{1;N},\ k\neq l.$$

It is also known here that (26) and (27) have their own error (also obtained on the basis of the Taylor series).

Similarly, it is possible to build a multivariate nonlinear regression model, that is, when statistical data are presented in the form of:

$$\overline{x}=\begin{pmatrix}x_{11} & x_{12} & \cdots & x_{1s}\\ x_{21} & x_{22} & \cdots & x_{2s}\\ \cdots & \cdots & \ddots & \vdots\\ x_{M1} & x_{M2} & \cdots & x_{Ms}\end{pmatrix}, \quad (28)$$

where s is the number of factors that affect the observed quantity.

Obviously, the problems in solving a regression-type problem for a generalized nonlinear case with respect to the sought unknown parameter values are as follows:

- calculation of the first derivatives for the functional dependence of a species $F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)$ on unknown parameters (although this can be done not only analytically, but also numerically);
- calculation of the second derivatives for the functional dependence of the species $F\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)},x_i\right)$ on unknown parameters (although this can be done not only analytically, but also numerically);
- Choosing an initial approximation $\left(\beta_1^{(0)},\beta_2^{(0)},\cdots,\beta_N^{(0)}\right)$ is difficult;
- At each iteration, k you need to find $J^{-1}\left(\beta_1^{(k)},\beta_2^{(k)},\cdots,\beta_N^{(k)}\right)$.

In this case, further research will focus on methods and algorithms that have a stochastic component. The stochastic component makes it possible to find solutions to various applied problems quite accurately and quickly. 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 particle of the swarm is subject to fairly simple rules of behavior that take into account the local success of each individual and the global optimum of all individuals (or some set of neighbors) of the swarm.

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

$$v_i(n+1) = v_i(n) + \alpha_1\left(x_i^{best}(n) - x_i(n)\right)r_1 + \alpha_2\left(x^*(n) - x_i(n)\right)r_2, \quad (29)$$

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

$$x_i(n+1) = x_i(n) + v_i(n+1), \quad (30)$$

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;

$r_1 = (r_{11}; r_{12}; \ldots; r_{1M})$, $r_1 = (r_{21}; r_{22}; \ldots; r_{2M})$ 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 (29). The first term in (29) $v_i(n)$ 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 (30).

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)

$$F(\boldsymbol{x}) \rightarrow min, \ \boldsymbol{x} = (x_1, \ldots, x_M),$$

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 of the current population 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 $F\left(x_k\right) \le F\left(x_k^{best}\right)$, 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

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

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)

$$F(\boldsymbol{x}) \to min,\ \boldsymbol{x} = \left(x_1, \ldots, x_M\right),$$

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 $P=P\cup\left\{x_k\right\}, k=k+1$

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. $x'_{zj}=x_{lj}+\eta(n)\delta\left(x_j^{max}-x_j^{min}\right)\left(-1+2rand\left(\right)\right),$

$$j\in\overline{1;M}, z\in\overline{1;Z}$$

   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* $f\left(x'_{z^*}\right)<f\left(x_l\right)$ then $x_l=x'_{z^*}$.

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

7. Scout Bee Phase (Random Search):

$$x_{lj}=x_j^{min}+\left(x_j^{max}-x_j^{min}\right)rand\left(\right),$$
$$j\in\overline{1;M}, l\in\overline{L_s+1;K}$$

8. Stop Condition

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

9. Identify the bee with the best position

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

The result is x^*.

2.2.3. Differential evolution method

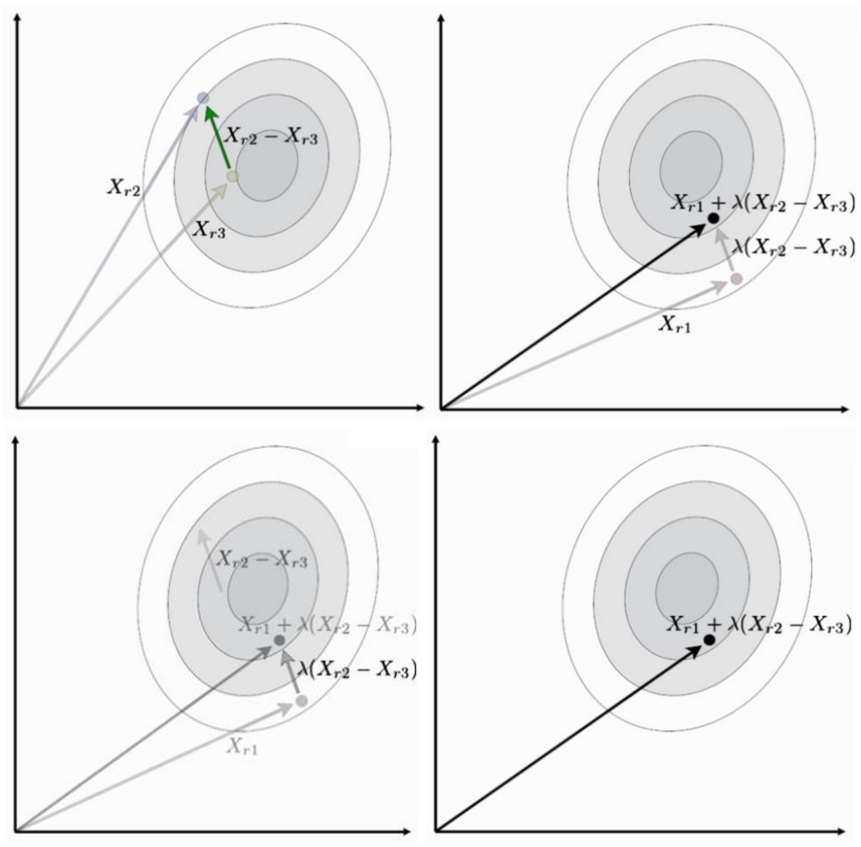
The main idea of the method of differential evolution is the combination of mutation and crossing to efficiently find optimal solutions in multidimensional parameter spaces. The method uses some ideas of genetic algorithms, but, unlike the latter, does not involve working with binary code.

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

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, $\overline{x_i} \neq x_1, \overline{x_i} \neq x_2, \overline{x_i} \neq x_3$, $x_1 \neq x_2 \neq x_3$.
2) A mutant vector is generated: $v = x_1 + F\left(x_2 - x_3\right)$, $F \in \left(0;2\right)$.
3) The vector difference $x_2 - x_3$ is scaled by a user-defined hyperparameter F, $F \in \left(0;2\right)$.

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



$X_{r2} - X_{r3}$
X_{r2}
X_{r3}
$X_{r1} + \lambda(X_{r2} - X_{r3})$
$\lambda(X_{r2} - X_{r3})$
X_{r1}
$X_{r2} - X_{r3}$
$X_{r1} + \lambda(X_{r2} - X_{r3})$
$\lambda(X_{r2} - X_{r3})$
X_{r1}
$X_{r1} + \lambda(X_{r2} - X_{r3})$

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

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 $r \in \left(0;1\right)$ according to the normal distribution.
   - If $r < P$, P is a given constant (another parameter of the algorithm $P \in \left(0;1\right)$), 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).

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. Of course, the number of calls to the procedure for finding the value of the target function in this case increases. 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. 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 $x_j^{Average}$ $\forall j = \overline{1;M}$ according to the following formula:

$$x_j^{Average} = \frac{1}{N}\sum_{i=1}^{N} x_{ij} \ \forall j = \overline{1;M}, \tag{31}$$

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:

$$x_{ij} = x_{ij} + h \cdot \frac{x_{Best,j} - x_{ij}}{\left\| x_{Best} - x_i \right\|},\ x_{Best} \neq x_i,\ i = \overline{1;N},\ j = \overline{1;M} \tag{32}$$

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

$$\eta(n) = \eta_{max}\alpha^{[n/K]}, \tag{33}$$

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.

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. Yavuz, G., Durmuş, B., & Aydın, D. (2022). Artificial bee colony algorithm with distant savants for constrained optimization. Applied Soft Computing, 116.
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. P., Liu, J. Ch., & Wei, Ch. (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. Fowler, M., Abbott, A. J., Murray, G. P., & McCall, P. J. (2021). Flying In-formation: A computational method for the classification of host seeking mosquito flight patterns using path segmentation and unsupervised machine learning. *bioRxiv*. https://doi.org/10.1101/2021.11.24.469809
6. Nayak, M., Das, S., Bhanja, U., & Senapati, M. R. (2023). Predictive Analysis for Cancer and Diabetes Using Simplex Method Based Social Spider Optimization Algorithm. *IETE Journal of Research*, *69*(10), 7342–7356. https://doi.org/10.1080/03772063.2022.2027276
7. Nadimi-Shahraki, M. H., Taghian, S., Mirjalili, S., Zamani, H., & Bahreinineja, A. (2022). Gaze cues learning-based grey wolf optimizer and its applications for solving engineering problems. *Journal of Computational Science*, 61. https://doi.org/10.1016/j.jocs.2022.101636
8. Yazdani, R., Mirmozaffari, M., Shadkam, E., & Taleghani, M. (2022). Minimizing total absolute deviation of job completion times on a single machine with maintenance activities using a Lion Optimization Algorithm. *Sustainable Operations and Computers*, *3*(3), 10–16. https://doi.org/10.1016/j.susoc.2021.08.003
9. Niu, G., Li, X., Wan, X., He, X., Zhao, Y., Yi, X., Chen, C., Xujun, L., Ying, G., & Huang, M. (2022). Dynamic optimization of wastewater treatment process based on novel multi-objective ant lion optimization and deep learning algorithm. *Journal of Cleaner Production*, 345. https://doi.org/10.1016/j.jclepro.2022.131140
10. Wang, S. H., Zhou, J., & Zhang, Y. D. (2022). Community-acquired pneumonia recognition by wavelet entropy and cat swarm optimization. *Mobile Networks and Applications*, 1–8. https://doi.org/10.1007/s11036-021-01897-0
11. Al-Dyani, W. Z., Ahmad, F. K., & Kamaruddin, S. S. (2022). Improvements of bat algorithm for optimal feature selection: A systematic literature review. *Intelligent Data Analysis*, *26*(1), 5–31. https://doi.org/10.3233/IDA-205455
12. Sun, B., Li, Y., Zeng, Y., Li, C., Shi, J., & Ma, X. (2021). Distribution transformer cluster flexible dispatching method based on discrete monkey algorithm. *Energy Reports*, 7, 1930–1942. https://doi.org/10.1016/j.egyr.2021.03.041
13. Rahkar Farshi, T., & Orujpour, M. (2021). A multi-modal bacterial foraging optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing*, *12*(11), 10035–10049. https://doi.org/10.1007/s12652-020-02755-9
14. Altay, O. (2022). Chaotic slime mould optimization algorithm for global optimization. *Artificial Intelligence Review*, *55*(5), 3979–4040. https://doi.org/10.1007/s10462-021-10100-5
15. Chakraborty, S., Saha, A. K., Sharma, S., Mirjalili, S., & Chakraborty, R. (2021). A novel enhanced whale optimization algorithm for global optimization. *Computers & Industrial Engineering*, 153. https://doi.org/10.1016/j.cie.2020.107086
16. 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.
17. 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
18. 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.
19. Castro, F., Vellido, A., Nebot, À., & Mugica, F. (2007) Applying Data Mining Techniques to e-Learning Problems. In L. C. Jain, R. A. Tedman & D. K. Tedman (Eds.), *Evolution of Teaching and Learning Paradigms in Intelligent Environment. Studies in Computational Intelligence*. Heidelberg: Springer, 62, 183–221. https://doi.org/10.1007/978-3-540-71974-8_8
20. Rahman, A., Sultan, K., Aldhafferi, N., & Alqahtani, A. (2018). Educational Data Mining for Enhanced teaching and learning. *Journal of Theoretical and Applied Information Technology*, *96*(14), 4417–4427.
21. Aldowah, H., Al-Samarraie, H., & Fauzy, W. M. (2019). Educational data mining and learning analytics for 21st century higher education: A review and synthesis. *Telematics and Informatics*, 37, 13–49. https://doi.org/10.1016/j.tele.2019.01.007

МЕТОДИ Й АЛГОРИТМИ РОЙОВОГО ІНТЕЛЕКТУ ДЛЯ ЗАДАЧ НЕЛІНІЙНОГО РЕГРЕСІЙНОГО АНАЛІЗУ ТА ОПТИМІЗАЦІЇ СКЛАДНИХ ПРОЦЕСІВ, ОБ'ЄКТІВ ТА СИСТЕМ: ОГЛЯД І МОДИФІКАЦІЯ МЕТОДІВ Й АЛГОРИТМІВ

Владислав Хайдуров[1,2]*, канд. техн. наук, ст. досл., https://orcid.org/0000-0002-4805-8880
Вадим Татенко[1], https://orcid.org/0009-0008-4869-9689
Микита Литовченко[1], https://orcid.org/0009-0001-6671-7763
Тамара Цюпій[3], канд. фіз.-мат. наук, доцент, https://orcid.org/0000-0003-2206-2897
Тетяна Жовновач[4], https://orcid.org/0000-0003-1037-4383
[1]Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Берестейський просп., 37, м. Київ, 03056, Україна;
[2]Інститут загальної енергетики НАН України, вул. Антоновича, 172, м. Київ, 03150, Україна;
[3]Національний університет біоресурсів і природокористування України, вул. Героїв Оборони, 15, м. Київ, 03041, Україна;
[4]Черкаська філія ПВНЗ «Європейський університет», вул. Смілянська, 83, м. Черкаси, 18008, 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. Нині існує багато сучасних методів й алгоритмів ройового інтелекту, які вирішують науково-прикладні завдання сьогодення, але вони потребують модифікацій у зв'язку з великими просторами пошуку оптимальних параметрів моделей. 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 задачі 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 обумовлена питаннями прискорення швидкості їх роботи під час розв'язання задач машинного навчання, зокрема у моделях нелінійної регресії, задачах класифікації, кластеризації, 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. Отримані модифікації відомих класичних ефективних методів й алгоритмів ройового інтелекту, які присутні у роботі, ефективно розв'язують складні науково-прикладні задачі конструювання складних об'єктів і систем. Порівняльний аналіз методів й алгоритмів буде проведено у наступному дослідженні за даною тематикою.*

Ключові слова: *оптимізація, ройовий інтелект, математичне моделювання, нелінійна регресія, складні об'єкти та системи.*

Надійшла до редколегії: 01.05.2024