Vol. 03, Iss. 1 (2026) 51-61, DOI: 10.61552/JSI.2026.01.006

JSDI

Journal of Sustainable Development Innovations

www.jsi.aspur.rs

An Inverse-Based Algorithm for Global Optimization

Ekaterina Gribanova[a,*], Dmitrij Leonov[a]

[a] *Tomsk state university of control systems and radioelectronics, Lenin Avenue 40, Tomsk, Russia.*

Keywords:

Global optimization, Multi-extreme function, Inverse problem, Neural networks, Metaheuristic methods

* Corresponding author:

Ekaterina Gribanova
E-mail: ekaterina.b.gribanova@tusur.ru

Received: 1 April 2025
Revised: 7 May 2025
Accepted: 4 June 2025

A B S T R A C T

Global function optimization is a vital area of applied mathematics, and it has numerous applications across fields such as machine learning, economics, and engineering design. Despite the extensive array of gradient-based and gradient-free optimization algorithms that leverage direct search strategies, the inverse approach to address optimization problems remains relatively unexplored. This paper focuses on the development of an inverse-based algorithms designed to solve global optimization problems that relies on the sequential resolution of inverse problems. The study involved executing computational experiments and comparing the proposed algorithm to existing methods to evaluate its effectiveness in optimizing the functions analyzed. In particular, the experiments focused on the global optimization of test functions and the training of neural networks. The obtained results can be used in further research and practical applications in diverse fields, including machine learning and the optimization of complex systems.

© 2026 Journal of Sustainable Development Innovations

1. INTRODUCTION

Global optimization is a significant domain in applied mathematics and computer science, and it is dedicated to discovering the best solution among all possible options [1]. This search helps improve the efficiency of resource use – such as time and productivity – in real-world applications. Global optimization problems are prevalent across a diverse range of fields, including engineering design, economics, machine learning, and operations research. Meanwhile, the complexity of optimization problems in science and industry is increasing [2], which results in optimized functions that are characterized by multimodality, nonlinearity, and intricate relief structures. The development of methods to address such challenges has become a pressing issue that numerous researchers are actively engaged in.

There are two primary approaches to addressing optimization problems, each of which has its own advantages and limitations. The first approach employs gradient methods that use the partial derivative values of the objective function to facilitate the optimization process. The gradient provides essential information regarding the fastest-descent direction, facilitating efficient discovery of the closest extremum. However, in the case of multi-extreme function, this solution method may lead to the identification of a local minimum, potentially bypassing the global minimum. Nonetheless, this method is widely employed, particularly in machine learning applications, because it efficiently finds solutions in large-dimensional spaces. For example, gradient descent and stochastic gradient algorithms are employed to train neural networks that consist of numerous adjustable parameters.

The second approach employs gradient-free metaheuristic methods. These methods perform an iterative exploration of the search space using heuristics to identify suboptimal solutions [3]. The collection of metaheuristic techniques is broad and constantly being enhanced with new variations because of the absence of a universally optimal method, as established by the no free lunch theorems [4]. Among the prominent categories of methods, we can identify evolutionary methods, swarm-based algorithms, approaches drawn from mathematics, chemistry, and physics, and human-inspired techniques [5]. Evolution-based methods are grounded in evolutionary concepts and employ mechanisms such as selection, interbreeding, and mutation to generate new populations [6]. Methods based on swarm intelligence replicate the social behaviors of various species in the environment—such as bees [7], wolves [8], hyenas [9], and birds [10] – often focusing on efficient food search. Another class of methods is founded on the principles of physical laws, chemical reactions, and mathematical models and encompasses techniques such as the sine-cosine algorithm [11], the arithmetic optimization algorithm [12], simulated annealing [13], and the gravitational search algorithm [14]. Human-based algorithms are predicated on observations of human behavior and activities, which are exemplified by concepts such as group learning [15] and student psychology [16]. Although metaheuristic methods are effective for solving complex structured problems, their performance typically decreases with increasing dimensionality, which presents significant challenges for their application to problem problems with numerous parameters. Nevertheless, in addition to incorporating gradients, optimization algorithms can differ in their strategies for generating new argument values using either forward or backward approaches. In the context of a direct strategy, new argument values are calculated based on previous argument values and their corresponding function value. In the application of the reverse strategy, new argument values are established such that the function value aligns with the target value. Most algorithms employ a direct strategy; however, the reverse strategy remains inadequately understood. The objective of this study was to develop an metaheuristic inverse-based algorithms to address the problem of global function optimization.

2. INVERSE-BASED ALGORITHM

Global optimization involves identifying the extremum–either minimum or maximum–of an objective function. In contrast to local optimization problems, which aim to identify a minimum or maximum within a specific neighborhood, global optimization seeks to determine the minimum or maximum across the entire domain of the function. Without compromising generality, we examine the minimization of the objective function, expressed as follows:

$$\min_{x \in R^m} f(x) \tag{1}$$

where $f(x)$ is the objective function. The assumptions include the continuity of the objective function and the existence of a global minimum for problem (1). In this study, we also assume that the function is differentiable.

A direct approach to solving optimization problems involves deriving new argument values x^* based on the information provided by the previous argument values x and their corresponding function value $f(x)$ (Figure 1). In the inverse approach, the argument values are calculated based on the target function value (Figure 2). Initially, the argument values are determined to obtain the function output of $f1$. Then, the function value is adjusted to $f2$, which requires the calculation of new argument values. Therefore, a single-point inverse problem is addressed for each iteration [17-18]:

$$f(x) = y \tag{2}$$

Here, y denotes a decreasing sequence of values with a positive step, which is defined as a y=y-step. Such problems (2) are deemed incorrect, and their solutions are obtained by applying l1 and l2 regularization methods [19] or expert evaluations.

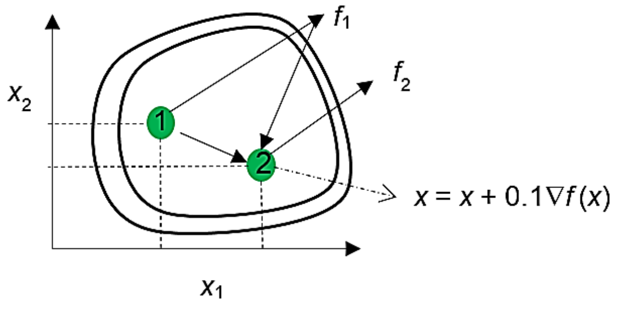

Fig. 1. Direct approach.

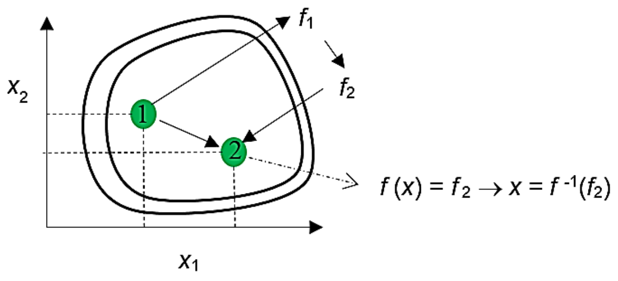

Fig. 2. Inverse approach.

The proposed algorithm Random Choice (RC) is based on the coordinate descent principle, where a single argument is selected for modification in each iteration. In this algorithm, the selection is determined by probability β, which applies equally to all arguments (Figure 3).

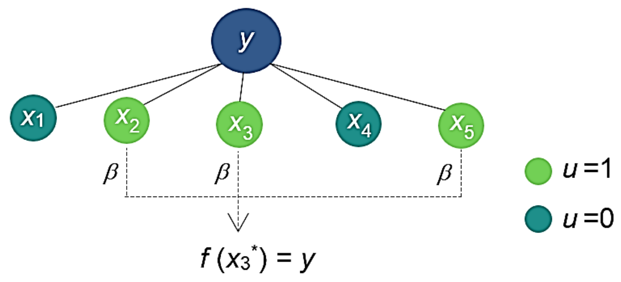

Fig. 3. Selection of the argument to modify.

However, if a priori information exists regarding the most effective methods for achieving the target value, this probability can be customized individually for each argument. Each argument of the function is associated with an indicator u, which characterizes its feasibility for inclusion in calculations. This indicator assumes one of two values: 1, indicating the feasibility of using an argument, and 0, indicating that the argument cannot be used. The algorithm is derived from the stochastic method for addressing the inverse problem detailed in [20] and has been adapted to obtain the global minimum of the function.

Algorithm RC

```
**Input**: Objective function $f(x)$, number of arguments $n$, step size for function modification $\Delta y$, ratio of step size reduction $r$, modifying the step reduction factor $q$, upper limit of the step reduction factor $r_{max}$, indicator of the possibility of use in calculations $u = 1$, maximum iterations $N$
**Output**: $x, f(x)$
Randomly generate an initial starting point $x$, $y = f(x)$
**for** $v = 1, N$ **do**
    $y_{last} = y,\ y = y - \frac{\Delta y}{r},\ h = \sum_{j=1}^{n} u_j$
    **for** $m = 1, n$ **do**
        **if** $u_m = 1$ **then** $\beta_m = \frac{1}{h}$ **else** $\beta_m = 0$
    **end**
    Randomly selecting argument $k$ according to probability $\beta$
    **if** $u_i = 0\ (\forall i = 1, n)$ **then**
        $r = r \cdot q$
        **if** $r > r_{max}$ **then exit**
        **else** $u_i = 1\ (\forall i = 1, n)$, $y = y_{last}$, **continuing to the next iteration in the loop**
    **end**
    Computing $x_k^*$ by resolving the equation $f(x_k^*) = y$
    **if** $(f(x_k^*) < y_{last})$ **then** $x_k = x_k^*$, $y = f(x_k^*)$, $u_i = 1\ (\forall i = 1, n)$ **else** $y = y_{last}, u_k = 0$
**end**
```

The second algorithm, developed under derivative choice (DC), calculates the partial derivatives for each argument and uses the normalized values of these derivatives as probabilities. Thus, the probability calculation ⍰ will be performed as follows:

$$c_m = \frac{\partial f}{\partial x_m}, \quad m = 1..n, \tag{3}$$

$$\beta_m = \frac{c_m \cdot u_m}{\sum_{j=1}^{n} c_m \cdot u_m}. \tag{4}$$

In this algorithm, an element of randomness is preserved in the selection of an argument while incorporating the influence of the derivative, which is beneficial for local minimum search.

3. EXPERIMENTS IN GLOBAL OPTIMIZATION OF TEST FUNCTIONS

This study examines a solution to the global optimization problem of the Rastrigin function in the range of -100 to 100. A distinctive characteristic of this function is the existence of numerous local minima and a singular global minimum located at the point $x = 0$, where the function attains a value of 0 (Fig. 4). To evaluate the performance of the developed algorithm against existing methodologies, we selected the following methods implemented in Python libraries:

- Basinhopping: a two-phase method that merges a global search algorithm with local minimization at each step.
- Particle Swarm Optimization algorithm (PSO): a computational method that optimizes a problem by iteratively adjusting a population of particles in the search space based on their position, velocity, and a quality measure.
- Genetic algorithm (GA): a metaheuristic inspired by evolution that optimizes problems using biological operators like selection, crossover, and mutation.
- Dual annealing (DA): a global optimization algorithm based on simulated annealing, where a temperature parameter balances exploration and convergence, enhanced by integrating local search methods.

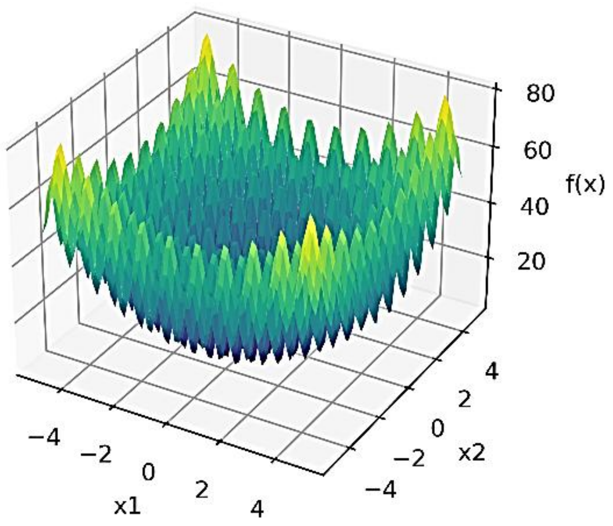

Fig. 4. The Rastrigin function.

The PSO and GA algorithms were implemented using the PySwarms and Deap libraries, respectively, and the DA and Basinhopping methods were implemented using the scipy.optimize module. The default hyperparameters were used for the Basinhopping, PSO, and DA algorithms. The following parameters were set for the genetic algorithm: population size = 500, number of generations = 700, crossover probability = 0.9, and mutation probability = 0.2. For the RC and DC methods, only the maximum value of the step reduction parameter rmax was modified: for the Rosenbrock function, it was set to 108, while for the other functions, it was set to 1018. The other hyperparameters were kept unchanged: the step size Δy was set to 100, the step reduction coefficient r was set to 1, and the modifying step reduction factor q was set to 2.

Table 1 presents the experimental results over 100 random realizations with the number of arguments of the Rastrigin function varying from 5 to 200. The evaluation criteria for the algorithms include the mean value of the function (Mean) and the average solution time t. According to the obtained results, the developed RC and DC algorithms demonstrated the best criterion values for 5 and 10 arguments. With an increase in the number of arguments, the algorithms demonstrated lower function values than the other algorithms but required more time. The best metaheuristic algorithm was DA because it achieved the lowest value of the optimized function. Figure 5 shows the success rate for the case with 10 arguments. A successful identification of the function minimum was defined as a solution in which the absolute deviation of the function value from the global minimum did not exceed 0.1.

Table 2 presents the results of solving optimization problems for various test functions included in the CEC 2022 Benchmark test set and described in [21]. The evaluation criteria include the minimum value of the objective function (Min), the mean value of the function (Mean), and the root mean square deviation of the function (STD). The first column lists the names of the functions and search intervals. The number of function arguments was set to 10. In some experiments, the genetic algorithm failed to solve the tasks; thus, the results were omitted. All methods successfully optimized the simple Zakharov function, which does not have local minima. For optimizing the Rosenbrock function in a complex landscape, the best results were achieved by basin hopping and DA. Among the functions with numerous local extremes, the developed RC and DC algorithms demonstrated the worst performance on the Drop-Wave function, which is highly multimodal and exhibits significant complexity.

Table 1. The results of optimizing the Rastrigin function.

The number of arguments	Metrics	Basinhopping	PSO	GA	DA	RC	DC
5	Mean	0.45	5.85	0.064	$3.2 \cdot 10^{-14}$	$3.14 \cdot 10^{-14}$	$1.77 \cdot 10^{-13}$
	t, sec.	0.37	0.25	11.72	0.59	0.19	0.19
10	Mean	2.09	29.85	0.157	0.02	$6.39 \cdot 10^{-12}$	$4.55 \cdot 10^{-13}$
	t, sec.	2.92	9.91	16.24	1.81	1.04	0.88
50	Mean	461.04	12,987.67	423.41	0.119	$7.69 \cdot 10^{-10}$	$4.5 \cdot 10^{-11}$
	t, sec.	86	1.2	52.2	7.98	52.5	45.5
100	Mean	255.70	56,407.38	1,047.9	0.3	$1.24 \cdot 10^{-9}$	$1.47 \cdot 10^{-9}$
	t, sec.	933	2.07	96.4	18.9	403.8	284.4
200	Mean	18,286.25	170,645.83	158,577.07	0.2	$9.96 \cdot 10^{-9}$	$5.39 \cdot 10^{-9}$
	t, sec.	3,431	4	191	51.3	1,997	1,606

Table 2. The results of optimizing test functions.

Function	Metrics	Basinhopping	PSO	DA	RC	DC
Zakharov [−100, 100]	Min	$1.68 \cdot 10^{-16}$	$1.13 \cdot 10^{-7}$	$1.42 \cdot 10^{-14}$	$1.18 \cdot 10^{-14}$	$5.68 \cdot 10^{-14}$
	Mean	$4.47 \cdot 10^{-15}$	1.97	0.02	$1.37 \cdot 10^{-14}$	$1.54 \cdot 10^{-12}$
	STD	$2.43 \cdot 10^{-15}$	5.44	0.14	$1.2 \cdot 10^{-15}$	$2.3 \cdot 10^{-12}$
Rosenbrok [−3, 3]	Min	$3.58 \cdot 10^{-11}$	4.33	$2.35 \cdot 10^{-11}$	$3.6 \cdot 10^{-3}$	$1.6 \cdot 10^{-3}$
	Mean	$5.64 \cdot 10^{-11}$	4058.84	0.08	0.37	0.29
	STD	$3.64 \cdot 10^{-12}$	14056.7	0.56	0.78	0.55
Ackley [-33, 33]	Min	1.15	$5.75 \cdot 10^{-6}$	$1.02 \cdot 10^{-8}$	$1.82 \cdot 10^{-14}$	$1.46 \cdot 10^{-14}$
	Mean	18.86	1.02	$1.82 \cdot 10^{-8}$	1.42	1.69
	STD	2.63	0.95	$2.92 \cdot 10^{-9}$	3.56	3.76
Griewank [−600, 600]	Min	$1.88 \cdot 10^{-11}$	20	$1.19 \cdot 10^{-11}$	0	$5.69 \cdot 10^{-4}$
	Mean	0.21	20.17	0.03	0.06	0.07
	STD	0.99	0.12	0.03	0.04	0.04
Schwefel [−500, 500]	Min	473.80	750.21	$1.27 \cdot 10^{-4}$	$1.27 \cdot 10^{-4}$	$1.27 \cdot 10^{-4}$
	Mean	1593.66	1461.23	$1.27 \cdot 10^{-4}$	353.64	364.61
	STD	516.47	319.91	$5.26 \cdot 10^{-9}$	248.04	234.5
Alpine 1 [−10, 10]	Min	$1.52 \cdot 10^{-7}$	$2.2 \cdot 10^{-4}$	$4.88 \cdot 10^{-9}$	$6.61 \cdot 10^{-7}$	$5.87 \cdot 10^{-7}$
	Mean	0.95	0.21	$4.87 \cdot 10^{-5}$	$5.53 \cdot 10^{-6}$	$5.81 \cdot 10^{-6}$
	STD	3.03	0.33	$6.84 \cdot 10^{-5}$	$3.21 \cdot 10^{-6}$	$3.19 \cdot 10^{-6}$
Drop-Wave [−5.12, 5.12]	Min	$1.71 \cdot 10^{-14}$	0.06	$8.88 \cdot 10^{-15}$	0.71	0.52
	Mean	0.64	0.26	0.28	0.89	0.9
	STD	0.43	0.11	0.15	0.06	0.07

Thus, the results of the computational experiments demonstrate the feasibility of employing the proposed algorithms to tackle global optimization challenges. These algorithms offer several advantages, including ease of implementation stemming from their independence from complex concepts such as selection in genetic algorithms or temperature in the simulated annealing algorithm. Furthermore, the proposed algorithms are memory-efficient as they do not require storing object sets, such as particles, or populations. Moreover, these algorithms do not require prior specification of the search interval, which may often be uncertain; instead, they suffice to define only the initial starting point to obtain a solution. One limitation of the proposed algorithms is that it relies on identifying the function root during execution, which is critical because the efficiency of the root-finding process directly impacts the optimization outcome. In addition, modifying only one argument per iteration may complicate the solution-finding process, especially for functions exhibiting regions with gradual variations or intricate landscapes. Therefore, the proposed algorithms can be effectively applied to specific tasks where they deliver optimal performance. Particularly, they achieved the best results when tested on the Rastrigin function.

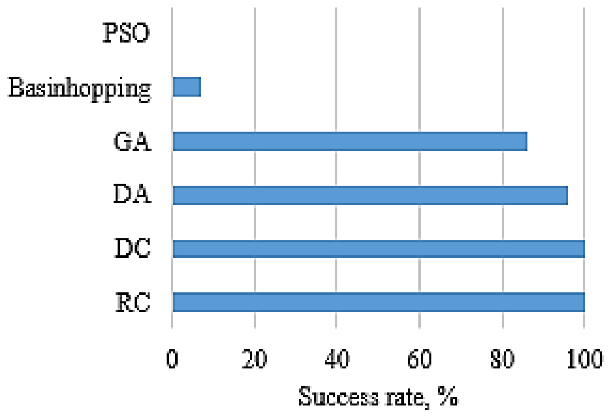

Fig. 5. Success rate in optimizing the Rastrigin function.

4. THE NEURAL NETWORK TRAINING ALGORITHM

One of the main trends in machine learning is artificial neural networks, which mimic the functioning of the human brain and possess the ability to autonomously extract features from data, making them a powerful tool for processing complex information streams. Applications based on neural networks recognize human speech, detect fraud, control autonomous vehicles and traffic [22], and predict economic indicators .

Training a neural network is an optimization problem characterized by several trainable weight parameters and components forming the objective function, as well as the presence of many local minima. The classical approach to solving this problem is gradient descent and its various modifications: stochastic gradient descent and adaptive gradient descent (e.g., Adagrad, Adadelta, RMSProp, Adam, etc.), and their improved variants. However, these methods have the following limitations: high dependence on the initially generated weight values and the risk of getting stuck at the local minimum. To address these shortcomings, previous studies have explored the application of metaheuristic algorithms, such as genetic algorithms [23], simulated annealing, and others. However, due to the large number of parameters to be optimized, solving a problem using heuristic algorithms requires significant computational resources and can also be inefficient. Thus, the development of new methods for training neural networks and their subsequent investigation is an important task. This study examines the application of the developed RC and DC algorithms to the training of neural networks.

The structure of the neural network includes the input data, hidden layer neurons, output neurons, and desired weight coefficients (Figure 6). The bias is also accounted for in the calculations by adding a column of ones to both feature array x and hidden layer array l. This study examines experiments conducted using a single-layer neural network and a neural network with one hidden layer, which is frequently used in data analysis [24].

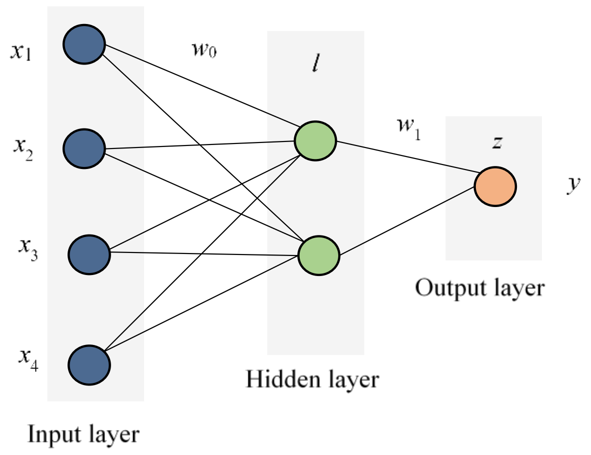

Fig. 6. Neural network structure.

The function to be optimized is the error function *J*, which reflects the deviation of predicted values from actual values:

$$J(w) = \sum_{i=1}^{n} (z_i - y_i)^2, \tag{5}$$

where *y* is the vector of actual output values; *z* is the vector of predicted values; and *n* is the number of observations.

To optimize the parameters of the neural network, the RC algorithm includes the following steps.

Step 1. Randomly generate weights *w* and compute the error function:

$$J_{prev} = J(w). \tag{6}$$

Step 2. Reduce the error function by a step size:

$$J^* = J_{prev} - \frac{\Delta J}{r} \tag{7}$$

Step 3. Select the weight coefficient w_{kj} (where k is the layer number and j is the neuron number in the layer) according to the probabilities β. If no coefficient is selected, the step reduction factor is increased: $r = r \cdot q$. If r exceeds $r_{\max}$, the algorithm terminates; otherwise, for all weights w, the usage indicator u is set to 1, and the process returns to Step 2.

Step 4. Modify the selected weight w_{kj} by solving the following equation:

$$J(w_{kj}^*) = J^* \tag{8}$$

In this process, if Newton's method is applied, the gradients are calculated as follows:

$$\begin{aligned} g_0 &= X^T l_{delta}, \\ g_1 &= l^T z_{delta}. \end{aligned} \tag{9}$$

Where the error at the output layer is calculated as (f' is the derivative of the activation function, f_2 is the activation function of the output layer):

$$z_{error} = z - y, \tag{10}$$

$$z_{delta} = z_{error} \odot f_2'(z), \tag{11}$$

where $\odot$ is the element-wise multiplication.

The error at the hidden layer can be defined as (f_1 is the activation function of the first layer):

$$l_{error} = z_{delta} \cdot w_1^T, \tag{12}$$

$$l_{delta} = l_{error} \odot f_1'(l). \tag{13}$$

For the second-level weights, the iterative adjustment in accordance with Newton's method is performed using the following formula:

$$w^*_{1j} = w_{1j} - \frac{J(w) - J^*}{2g_{1j}}. \tag{14}$$

For the first-level weights, the formula is as follows:

$$w^*_{0j} = w_{0j} - \frac{J(w) - J^*}{4g_{0j}}. \tag{15}$$

Step 5. Compute the new error function value $J_{new} = J(w^*)$, If $J_{new}<J_{prev}$, then $w=w^*$, u=1 for all weights, and $J_{prev}= J_{new}$. Proceed to Step 2. Otherwise, set u_{kj} = 0. Proceed to Step 2.

In the case of applying the RC algorithm, the probabilities in Step 3 are equal; when using the DC algorithm, they are proportional to the absolute values of the partial derivatives.

5. COMPUTATIONAL EXPERIMENTS ON NEURAL NETWORK TRAINING

5.1 Single-layer neural network

To conduct computational experiments with a single-layer neural network, data from the KEEL repository were used, including datasets such as treasury, stock, autoMPG8, baseball, concrete, and machineCPU. The input and output variables were standardized during preprocessing. Both the classical activation function (linear) and more complex ones were considered, including those proposed for time series modeling [25]: snake, sincos, ISRU, actg, ln, softsign, th, softplus, and cloglogm. In addition, the Himmelblau function, which is widely used as a benchmark in global optimization tasks, was also considered an activation function.

As part of the study, two developed optimization algorithms, RC and DC, were considered, along with classical methods widely used for neural network training, such as Adam and SGD. Additionally, the metaheuristic approach DA was applied. The experiments were conducted over 10 random runs. For the Adam and SGD methods, the learning rate was varied from 0.00001 to 1, and the best option was selected. For the RC and DC algorithms, the hyperparameters remained unchanged: specifically, the step ΔJ was equal to the initial value of the function J divided by 10, the maximum step reduction parameter rmax was 100,000, the step reduction coefficient r=1, and the modifying step reduction factor q=2. The initial weight coefficient values were generated within the range [0, 0.01].

Based on the results presented in Table 3, the following conclusions can be drawn. DA demonstrated one of the best performances for solving global optimization problems; however, in most scenarios, it falls behind other methods when training neural networks. The best performance achieved by this method was obtained using the sincos activation function. Furthermore, for this activation function, the DC and RC methods demonstrated higher efficiency than the Adam and SGD. The DC method achieved the best results in most scenarios; however, in terms of computation time, both RC and DC were outperformed by the Adam and SGD methods (Figure 7). The comparison of the mean squared error (MSE) and mean absolute error (MAE) values (Table 4) reveals that the developed algorithms provided superior results for these metrics in the following datasets: concrete, AutoMPG8, and machineCPU.

Table 3. Number of best cases with different activation functions.

Dataset	The number of activation functions for which the method demonstrated the best results in terms of MSE					The number of activation functions for which the method demonstrated the best results in terms of MAE				
	Adam	SGD	RC	DC	DA	Adam	SGD	RC	DC	DA
treasury	1	0	2	7	1	2	0	0	8	1
stock	0	0	7	3	1	1	0	7	2	1
autoMPG8	0	1	3	7	0	0	0	5	6	0
baseball	0	3	4	3	0	0	3	4	3	0
concrete	1	0	2	7	1	1	0	5	5	0
machineCPU	3	5	1	2	0	1	1	4	5	0

Table 4. The best results in terms of MSE.

Dataset	Adam		SGD		RC		DC		DA	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
treasury	**$6.2\cdot10^{-4}$**	**$4.69\cdot10^{-3}$**	$9\cdot10^{-4}$	$6.1\cdot10^{-3}$	$6.5\cdot10^{-4}$	$5\cdot10^{-3}$	$6.5\cdot10^{-4}$	$4.7\cdot10^{-3}$	0.005	0.048
stock	0.0128	**0.0273**	0.013	0.028	**0.01278**	0.0274	0.0128	0.0274	0.036	0.158
autoMPG8	0.0183	0.0319	0.0178	0.031	0.0175	0.0301	**0.0174**	**0.03**	0.023	0.108
baseball	0.0378	0.0484	**0.0376**	**0.0482**	0.0384	0.0486	0.0386	0.0487	0.765	0.7
concrete	0.0343	0.0463	0.0342	0.0463	0.0336	0.0453	**0.0335**	**0.0452**	0.02	0.113
machineCPU	0.0421	0.0365	0.042	0.0365	0.042	0.0364	**0.0418**	**0.0363**	0.427	0.587

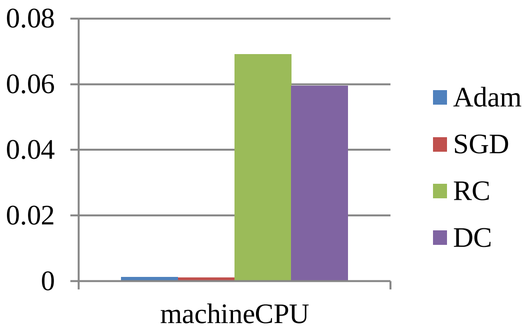

Fig. 7. The solution time for the machineCPU dataset.

The DA, RC, and DC algorithms demonstrated more efficient model training with the sincos activation function than classical methods; thus, the possibility of its application to the problem of time series modelling was considered. The temperature time-series from an Irish weather station [26] was used as the test dataset. The dataset includes 6075 daily observations from August 11, 2003, to May 31, 2020, with the last 1500 observations used as the test set. The following data were used to train the neural network:

- Output variable: maximum temperature (Maxtp).
- Input variables: cbl – mean corrected barometer level pressure (hPa), wdsp – mean wind speed (kt), hm – highest ten-minute mean wind speed (kt), maxtp_s – maximum temperature at the previous time step.

The initial weights were generated within the range from 0 to 1.

Figure 8 illustrates the actual values for the test dataset and the predicted values obtained using the RC algorithm. Figure 9 presents a boxplot of the MSE values, which was constructed based on the test dataset for 100 random realizations. The RC method had the lowest variability and mean values. At the same time, the greatest variability was observed for the Adam and DA methods.

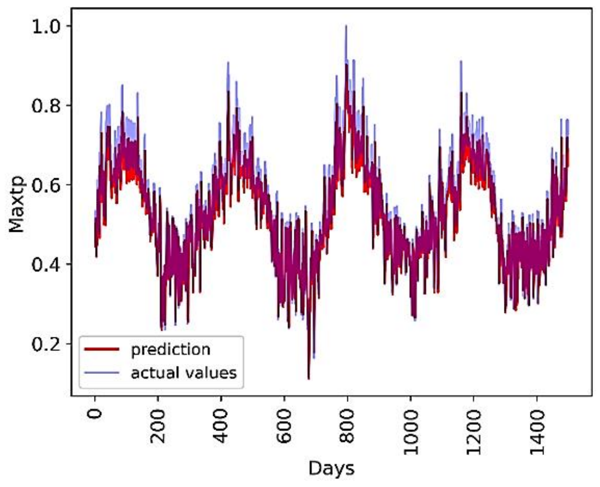

Fig. 8. Actual and predicted values of the test dataset (RC algorithm).

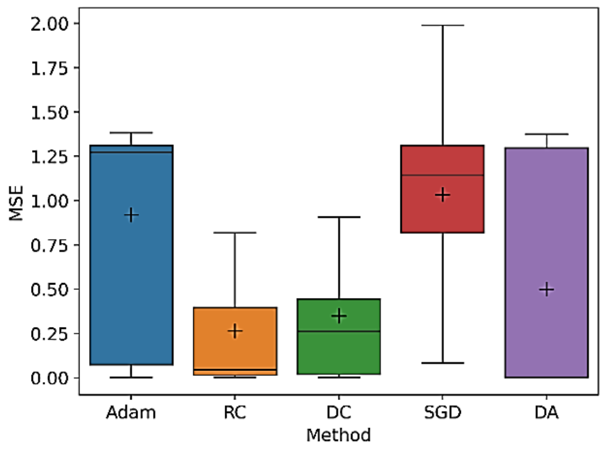

Fig. 9. Boxplots of MSE for various methods on the test sample. Hereinafter, plus—mean value, line—median, rectangle—25–75% quartile range, whiskers—minimum and maximum values or 1.5 interquartile range.

5.2 A neural network with a single hidden layer

A more complex neural network structure with a single hidden layer containing two neurons was also investigated. The activation function of the hidden layer was sincos, and the output layer used the Rastrigin function with normalization set to 5000.

The data for the experiments were based on the Wilshire 2500 Total Market Index (https://www.wilshireindexes.com), a key indicator of the U.S. stock market that reflects the dynamics of mid-cap stocks. For training, we utilized daily data from 2021-01-04 to 2023-12-29, comprising 754 points. The last 200 observations were used as the test set. The following data were used for training the neural network:

- Output variables: Wilshire index values on day *t*.
- Input variables: Wilshire index values on days *t*-1, *t*-2, *t*-3, and *t*-4.

The BoxPlot for MSE on the test dataset is presented in Figure 10, which shows that the RC and DC methods demonstrate the smallest variance as well as the lowest mean and median values. It is noteworthy that the DA method was the most time-consuming, with a runtime of 46.8 s to solve the problem. For the RC, DC, Adam, and SGD algorithms, the average runtimes were 4.83, 3.19, 2.77, and 1.53 s, respectively.

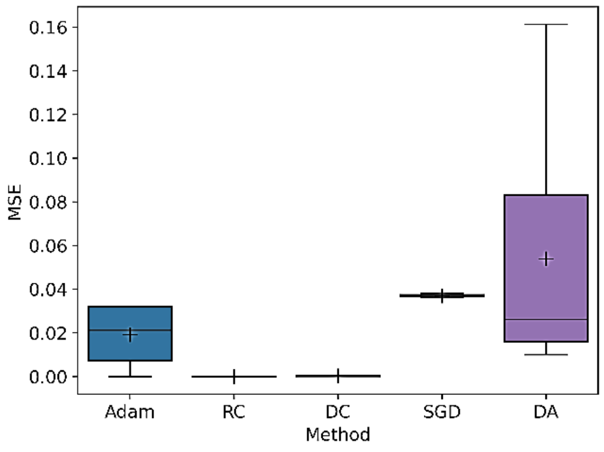

Fig. 10. Boxplots of MSE for various methods on the test sample.

6. CONCLUSION

In this paper, we have proposed an inverse-based algorithms designed to address the problem of global optimization of a function using a coordinate descent method that incorporates a stochastic selection of arguments. The proposed algorithms were applied to minimize the various test functions, thereby demonstrating their effectiveness compared with the established methods on specific tasks. The developed algorithms can be effectively applied to solve Rastrigin function optimization problems, as well as in neural network training scenarios that utilize complex activation functions like sincos or Rastrigin. Future work will focus on applying the proposed algorithms in the field of machine learning to comprehensively evaluate their effectiveness in typical tasks encountered in this domain.

Acknowledgement

This study was funded by the Russian Science Foundation, grant number 25-21-00123 “Application of methods for solving inverse ill-posed problems in machine learning”: https://rscf.ru/project/25-21-00123/

REFERENCES

[1] L. Wu, H. Chen, and Z. Yang, “A neural network transformation based global optimization algorithm,” *Inf. Sci.*, vol. 694, p. 121693, 2025, doi: 10.1016/j.ins.2024.121693.

[2] B.-W. Xiang, Y.-X. Xiang, and T.-Y. Zhang, “Rabbit algorithm for global optimization,” *Appl. Math. Model.*, vol. 140, p. 115860, 2025, doi: 10.1016/j.swevo.2024.101779.

[3] M. Ghazaan et al., “A novel adaptive optimization scheme for advancing metaheuristics and global optimization,” *Swarm Evol. Comput.*, vol. 91, 101779, 2024, doi: 10.1016/j.swevo.2024.101779.

[4] D. Wolpert and W. Macready, “No Free Lunch Theorems for Optimization,” *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997, doi: 10.1109/4235.585893.

[5] S. Biswas et al., “Integrating Differential Evolution into Gazelle Optimization for advanced global optimization and engineering applications,” *Comput. Methods Appl. Mech. Eng.*, vol. 434, 117588, 2025, doi: 10.1016/j.cma.2024.117588.

[6] M. Bumin and M. Ozcalici, “Predicting the direction of financial dollarization movement with genetic algorithm and machine learning algorithms: The case of Turkey,” *Expert Syst. Appl.*, vol. 213, 119301, 2023, doi: 10.1016/j.eswa.2022.119301.

[7] X. Chu et al., “An artificial bee colony algorithm with adaptive heterogeneous competition for global optimization problems,” *Appl. Soft Comput.*, vol. 93, 106391, 2020, doi: 10.1016/j.asoc.2020.106391.

[8] R. Rajakumar, K. Sekaran, C.-H. Hsu, and S. Kadry, “Accelerated grey wolf optimization for global optimization problems,” *Technol. Forecast. Soc. Change*, vol. 169, p. 120824, 2021, doi: 10.1016/j.techfore.2021.120824.

[9] G. Dhiman and V. Kumar, “Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications,” *Adv. Eng. Softw.*, vol. 114, pp. 48–70, 2017, doi: 10.1016/j.advengsoft.2017.05.014.

[10] H. Zamani, M. Nadimi-Shahraki, and A. Gandomi, “CCSA: Conscious Neighborhood-based Crow Search Algorithm for solving global optimization problems,” *Appl. Soft Comput.*, vol. 85, p. 105583, 2019, doi: 10.1016/j.asoc.2019.105583.

[11] S. Mirjalili, “SCA: A Sine Cosine Algorithm for solving optimization problems,” *Knowl.-Based Syst.*, vol. 96, pp. 120–133, 2016, doi: 10.1016/j.knosys.2015.12.022.

[12] L. Abualigah et al., “The arithmetic optimization algorithm,” *Comput. Methods Appl. Mech. Eng.*, vol. 376, 113609, 2021, doi: 10.1016/j.cma.2020.113609.

[13] D. Delahaye, S. Chaimatanan, and M. Mongeau, “Simulated Annealing: From Basics to Applications,” in *Handbook of Metaheuristics*, Springer, Cham, 2019, pp. 1–33.

[14] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, “GSA: A gravitational search algorithm,” *Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, 2009, doi: 10.1016/j.ins.2009.03.004.

[15] Y. Ziying and J. Zhigang, “Group Teaching Optimization Algorithm: A novel metaheuristic method for solving global optimization problems,” *Expert Syst. Appl.*, vol. 148, p. 113246, 2020, doi: 10.1016/j.eswa.2020.113246.

[16] B. Das, V. Mukherjee and D. Das, “Student psychology-based optimization algorithm: A new population-based optimization algorithm for solving optimization problems,” *Adv. Eng. Softw.*, vol. 146, 102804, 2020, doi: 10.1016/j.advengsoft.2020.102804.

[17] B. Gribanova, “Development of iterative algorithms for solving the inverse problem using inverse calculations,” *Eastern-European J. Enterp. Technol.*, vol. 4, no. 3, pp. 27–34, 2020, doi: 10.15587/1729-4061.2020.205048.

[18] E. B. Gribanova, “Algorithm for solving the inverse problems of economic analysis in the presence of limitations,” *EUREKA: Phys. Eng.*, vol. 1, pp. 70–78, 2020, doi: 10.21303/2461-4262.2020.001102.

[19] N. Ye, F. Roosta-Khorasani, and T. Cui, “Optimization Methods for Inverse Problems,” *MATRIX Annals*, vol. 2, pp. 121–140, 2017.

[20] E. B. Gribanova, “Stochastic algorithms to solve the economic analysis inverse problems with constraints,” *Proc. TUSUR Univ.*, vol. 19, no. 4, pp. 112–116, 2016, doi: 10.21293/1818-0442-2016-19-4-112-116.

[21] V. Plevris and G. Solorzano, “A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking,” *Data*, vol. 7, no. 4, p. 46, 2022, doi: 10.3390/data7040046.

[22] M. Wallner and T. Peráček, "Predictors of average wait time at an intersection using artificial neural network," *Int. J. Qual. Res.*, vol. 17, no. 3, pp. 867–890, 2023, doi: 10.24874/IJQR17.03-15.

[23] S. Ding, C. Su, and J. Yu, "An optimizing BP neural network algorithm based on genetic algorithm," *Artif. Intell. Rev.*, vol. 36, pp. 153–162, 2011, doi: 10.1007/s10462-011-9208-z.

[24] A. Xu, H. Chang, Y. Xu, R. Li, X. Li, and Y. Zhao, "Applying artificial neural networks (ANNs) to solve solid waste-related issues: A critical review," *Waste Manag.*, vol. 124, pp. 385–402, 2021, doi: 10.1016/j.wasman.2021.02.029.

[25] G. S. Gomes et al., "Comparison of new activation functions in neural network for forecasting financial time series," *Neural Comput. Appl.*, vol. 20, pp. 417–439, 2011, doi: 10.1007/s00521-010-0407-3.

[26] L. Ziyin et al., "Neural networks fail to learn periodic functions and how to fix it," in Proc. 34th Conf. *Neural Inf. Process. Syst.* (NeurIPS 2020), Vancouver, Canada, pp. 1–12, 2020.