Methods of Parametric Identification Based on Monkey Behavior

Eugene Fedorov, Anait Karapetyan, Kostiantyn Rudakov and Anatolii Chepynoha

Cherkasy State Technological University, Cherkasy, Shevchenko blvd., 460, 18006, Ukraine

Abstract

In modern conditions, a relevant task is the development of methods for the parametric identification of models of artificial neural networks used in intelligent computer systems. To enhance the effectiveness of parametric identification, metaheuristic methods based on monkey behavior (Monkey Search, Monkey Algorithm, Spider Monkey Optimization) have been proposed. These metaheuristic methods use dynamic parameters to ensure global search in the initial iterations and local search in the final iterations, which improves the accuracy of the search and does not require the transformation of the objective function. The proposed metaheuristics expand the application domain of monkey-based parametric identification methods and contribute to the improvement of the efficiency of intelligent computer systems. Further research perspectives include exploring the proposed methods for a wide range of artificial intelligence tasks.

Keywords

Metaheuristics, monkey search with dynamic parameters, monkey algorithm with dynamic parameters, spider monkey optimization with dynamic parameters, and parametric identification of artificial neural network models

1. Introduction

In modern conditions, the development of methods for the parametric identification of artificial neural network models used in intelligent computer systems [1-3] is an urgent task.

However, existing parametric identification methods that obtain approximate solutions through global search do not guarantee convergence, while methods that obtain approximate solutions through local search have a high probability of getting trapped in local optima. Furthermore, methods that obtain exact solutions have high computational complexity. Thus, there is a problem of insufficient efficiency in existing parametric identification methods.

To reduce the probability of getting stuck in a local extremum and to speed up parametric identification, modern heuristics (or metaheuristics) are used [4-5]. Metaheuristics expand the possibilities of heuristics by combining heuristic methods based on a high-level strategy [6-7]. Metaheuristics often use the behavior of one or a group of animals [8-9]. Metaheuristics are an approximate and usually non-deterministic method [10-11]. Advanced metaheuristics use experience accumulated during the search, represented in memory, to manage the search process [12-13].

Object of research: The process of parametric identification of an artificial neural network model. Subject of research: Methods of parametric identification of an artificial neural network model based on metaheuristics. This work aims to enhance the efficiency of the parametric identification of an artificial neural network model using metaheuristic methods with dynamic parameters based on the behavior of monkeys. To achieve the stated objective, the following tasks need to be addressed:

1. To develop a numerical optimization method based on the synthesis of the monkey search algorithm.

International Scientific Symposium «Intelligent Solutions» IntSol-2023, September 27–28, 2023, Kyiv-Uzhhorod, Ukraine
EMAIL: anait.r.karapetyan@gmail.com (A. Karapetyan); y.fedorov@chdtu.edu.ua (E. Fedorov); k.rudakov@chdtu.edu.ua (K. Rudakov); toxachep@ukr.net (A. Chepynoha)
ORCID: 0000-0002-7412-3252 (A. Karapetyan); 0000-0003-3841-7373 (E. Fedorov); 0000-0003-0000-6077 (K. Rudakov); 0000-0003-3921-6557 (A. Chepynoha)

2. To develop a numerical optimization method based on the synthesis of the monkey algorithm.
3. To develop a numerical optimization method based on the synthesis of the spider monkey optimization algorithm.
4. Conduct a numerical investigation of the proposed optimization methods.

2. Literature Review

Existing metaheuristics are categorized into non-nature-inspired, evolutionary, immune, biological, physical, chemical, mathematical, and social. Almost all metaheuristics employ random search, which reduces the likelihood of converging to a random extremum. Metaheuristics are capable of solving problems of discrete optimization (e.g., the traveling salesman problem, knapsack problem, assignment problem, clustering) and continuous optimization (finding the point at which a function reaches an extremum).

Existing metaheuristics have one or more of the following drawbacks:

- failure to consider the impact of iteration number on the solution search process [14-15];
- the method's description is oriented towards solving only specific problems or is an abstract method description [16-17];
- lack of guaranteed convergence of the method [18-19];
- inability to use non-binary potential solutions [20-21];
- insufficient accuracy of the method [22-23];
- inability to solve problems of constrained optimization [24-25];
- lack of automated procedure for determining parameter values [26-27].

Therefore, the task arises to develop effective metaheuristic optimization methods. Among the most popular are metaheuristics based on monkey behavior [28].

3. Monkey search with dynamic parameters

The "monkey search" was proposed by Mucherino [28] and is based on the behavior of a monkey that climbs a tree in search of food, while remembering where it found good food in the past. Figure 1 shows a binary decision tree, with the path to the best node highlighted in bold. In this algorithm, the monkey randomly climbs the tree and discovers new left and right nodes. Node discovery corresponds to generating new solutions based on the current or current and best solution using perturbation. The monkey stops when the current best node (best solution) is found or when the maximum tree height is reached. Upon reaching either of these stopping conditions, the monkey descends back to the root node of the tree.

Then the monkey climbs up the tree again, choosing between previously discovered left and right nodes, with a higher probability of selecting the best nodes. A new binary tree is created at each iteration. When the number of nodes at the maximum tree level reaches a specified value, the construction of the current tree is completed. This algorithm uses memory that stores the best solutions obtained from the first trees. The first trees have randomly generated solutions as their roots, and subsequent trees have roots of solutions from memory, randomly selected. After filling all memory with the best solutions, if a new solution is better than one or more solutions in memory, then the new solution replaces the worst solution in memory. If the new solution is too close to a solution in memory, the new solution is ignored. The monkey search stops when the maximum number of built trees is reached. In contrast to the original method, this method uses dynamic mean square deviation to generate the position vector.

3.1. Algorithm for optimizing numerical functions

1. Initialization.
   1.1. Specifying the minimum distance between vertex positions (solutions) in memory ε, the parameter δ that controls the step size for generating a new solution, subject to the condition that, $\varepsilon > 0$, $0 < \delta < 1$.

1.2. Specifying the maximum number of trees N, maximum number of levels for each tree H, maximum number of vertices for each tree H at a given level N^H, memory size L, length of the vertex position vector M, minimum and maximum values for the solution $x_j^{\min}, x_j^{\max}$, $j \in \overline{1,M}$.

1.3. Specifying the cost function (objective function) $F(x) \to \min\limits_x$, where x is the vector position of the tree vertex.

1.4. Creating the optimal solution randomly as a real vector

$$x^* = (x_1^*,...,x_M^*),\ x_j^* = x_j^{\min} + (x_j^{\max} - x_j^{\min})U(0,1), \quad (1)$$

where $U(0,1)$ - is the function returns a standard uniformly distributed random number.

1.5. Memory $P^M = \varnothing$.

2. Tree number $n = 1$.

3. Creation of the n-th tree.

3.1. Set of explored node positions $V_n = \varnothing$.

3.2. Set of edges $A_n = \varnothing$.

3.3. Set of node positions H at the level $V_n^H = \varnothing$.

3.4. Exploring the root of the tree.

3.4.1. If $\left|P^M\right| < L$, then - randomly create a root position, then the root position is randomly created.

$$x^{root} = (x_1^{root},...,x_M^{root}),\ x_j^{root} = x_j^{\min} + (x_j^{\max} - x_j^{\min})U(0,1), \quad (2)$$

3.4.2. If $\left|P^M\right| = L$, then from memory, P^M a vertex with a number is randomly selected $v = round(1 + (L-1)U(0,1))$, which becomes a root x^{root}, where $round()$ – is a function rounding a number to the nearest integer.

3.4.3. Add the root position to the set of explored node positions

$$V_n = V_n \bigcup \{x^{root}\}. \quad (3)$$

3.5. Set the position of the current tree vertex

$$x^{cur} = x^{root}. \quad (4)$$

3.6. Ascending to explored tree vertices.

3.6.1. Tree level number $m = 1$.

3.6.2. If $\neg\exists x^{left} \in V_n : (x^{cur}, x^{left}) \in A_n$ and $\neg\exists x^{right} \in V_n : (x^{cur}, x^{right}) \in A_n$, then go to step 3.7.

3.6.3. Calculate the probability of ascending to the left vertex

$$p = \frac{1/F(x^{left})}{1/F(x^{left}) + 1/F(x^{right})}. \quad (5)$$

3.6.4. Select the position of the current vertex

$$\lambda = U(0,1),\ x^{cur} = \begin{cases} x^{left}, & p < \lambda \\ x^{right} & p \ge \lambda \end{cases}. \quad (6)$$

3.6.5. If $m < H$, then $m = m + 1$, go to step 3.6.2, otherwise 3.5.

3.7. Ascending to unexplored tree vertices.

3.7.1. Create the position of the left vertex x^{left} by perturbation (e.g., mutation).

3.7.1.1. $\sigma_j(m) = \left(1 - \frac{m}{H}\right)\delta(x_j^{\max} - x_j^{\min}),\ x_j^{left} = x_j^{cur} + \sigma_j(m)N(0,1),\ j \in \overline{1,M}$,

where $N(0,1)$ – is a function returning a standard normally distributed random number.

3.7.1.2. $x_j^{left} = \max\{x_j^{\min}, x_j^{left}\},\ x_j^{left} = \min\{x_j^{\max}, x_j^{left}\},\ j \in \overline{1,M}$.

3.7.2. Create the position of the right vertex x^{right} by perturbation (e.g., mutation).

3.7.2.1. $\sigma_j(m) = \left(1 - \frac{m}{H}\right)\delta(x_j^{\max} - x_j^{\min})$, $x_j^{right} = x_j^{cur} + \sigma_j(m)N(0,1)$, $j \in \overline{1,M}$.

3.7.2.2. $x_j^{right} = \max\{x_j^{\min}, x_j^{right}\}$, $x_j^{right} = \min\{x_j^{\max}, x_j^{right}\}$, $j \in \overline{1,M}$.

3.7.3. If $F(x^{cur}) < \min\{F(x^{left}), F(x^{right})\}$, then go to step 3.7.1.

3.7.4. Add the vertex positions to the set of explored vertex positions

$$V_n = V_n \bigcup \{x^{left}, x^{right}\}. \quad (7)$$

3.7.5. Add the edges to the set of edges

$$A_n = A_n \bigcup \{(x^{cur}, x^{left}), (x^{cur}, x^{right})\}. \quad (8)$$

3.7.6. If $m < H$, then $m = m + 1$, go to step 3.5.

3.7.7. Add the vertex position to the set of vertex positions H at the level

$$V_n^H = V_n^H \bigcup \{x^{left}, x^{right}\}. \quad (9)$$

3.7.8. If $\left|V_n^H\right| < N^H$, then go to step 3.5.

4. Determine the best vertex position of n-th tree

$$x_n^{best} = \arg\min_{x \in V_n^H} F(x). \quad (10)$$

5. Add the best vertex position of n-th tree to the memory.

5.1. If $\left|P^M\right| = 0$, then $P^M = P^M \bigcup \{x_n^{best}\}$, go to step 6.

5.2. If $\left|P^M\right| > 0 \wedge \min_{x \in P^M} \left\|x - x_n^{best}\right\| < \varepsilon$, then go to step 6.

5.3. If $\left|P^M\right| < L$, then $P^M = P^M \bigcup \{x_n^{best}\}$, go to step 6.

5.4. Determine the worst position in the memory.

5.5. If $F(x^{worst}) \geq F(x_n^{best})$, then $P^M = P^M \setminus \{x^{worst}\}$, $P^M = P^M \bigcup \{x_n^{best}\}$.

6. If $n < N$, then $n = n + 1$, go to step 3, otherwise – stop.

7. Identification of the best global position

$$x^* = \arg\min_{x \in P^M} F(x). \quad (11)$$

The solution is x^*.

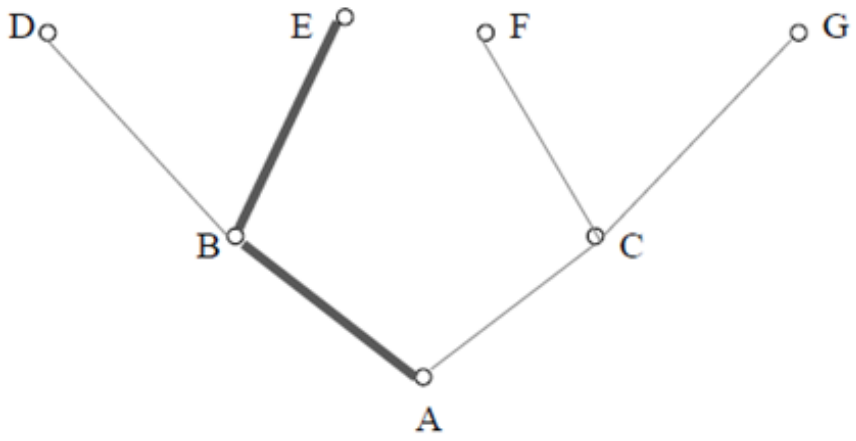

Figure 1: Binary decision tree

4. Monkey Algorithm with Dynamic Parameters

The Monkey Algorithm, proposed by Zhao and Tang [28], is based on the model of monkeys climbing mountains. It consists of three processes: the climbing process, the observation-jump process, and the somersault process. The climbing process is a step-by-step procedure of changing the positions of the monkeys from their initial positions to new positions, which uses a pseudogradient stochastic

approximation. After the climbing process, each monkey arrives at its vertex and enters the observation-jump process. It inspects its surroundings and determines if there are other positions higher than its current position. If so, it jumps from the current position and then repeats the climbing process until it reaches the mountain peak. After these processes, each monkey finds a local maximum mountain peak. The goal of the somersault process is to make the monkeys find globally maximum mountain peaks in new search areas. To achieve this, the monkeys perform a somersault into a new search area in the direction of the center of the current positions of all the monkeys. In contrast to the original method, this method uses dynamic step size to generate the position vector.

Algorithm for optimizing numerical functions

4.1. Algorithm for optimizing numerical functions

1. Initialization

1.1. Set the minimum and maximum step lengths $a^{\min}, a^{\max}$ for generating the position vector during the ascent process, and the minimum and maximum step lengths $b^{\min}, b^{\max}$ for generating the position vector during the observation-jump process, meanwhile $a^{\min} > 0, a^{\max} > 0, b^{\min} > 0, b^{\max} > 0$.

1.2. Set the maximum number of iterations N, the maximum number of iterations for local search N^L, the population size K, the length of the monkey's position vector M, and the minimum and maximum values for the position vector $x_j^{\min}, x_j^{\max}$, $j \in \overline{1,M}$.

1.3. Define the cost function (objective function):

$$F(x) \to \min_x, \tag{12}$$

where x is the monkey's position vector.

1.4. Create the initial population P.

1.4.1. Monkey number $k = 1$.

1.4.2. Generate a random position vector x_k

$$x_k = (x_{k1}, ..., x_{kM}), \;\; x_{kj} = x_j^{\min} + (x_j^{\max} - x_j^{\min})U(0,1). \tag{13}$$

where $U(0,1)$ – is a function that returns a standard uniformly distributed random number.

1.4.3. If $x_k \notin P$, then $P = P \bigcup \{x_k\}$, $k = k + 1$.

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

1.5. Determine the monkey with the best objective function value:

$$k^* = \arg\min_{k \in \overline{1,K}} F(x_k), \;\; x^* = x_{k^*}. \tag{14}$$

1.6. Set the initial step lengths $a(1) = a^{\max}$, $b(1) = b^{\max}$.

2. Number of iterations $n = 1$.

3. Climbing process.

3.1. Monkey number $k = 1$.

3.2. $m = 1$.

3.3. Create a position vector increment Δx_k

$$\Delta x_k = (\Delta x_{k1}, ..., \Delta x_{kM}), \;\; \Delta x_{kj} = round(1 - 2U(0,1))a(n), \tag{15}$$

where $round()$ – is a function that rounds a number to the nearest integer.

3.4. Calculate the pseudo-gradient of the cost function

$$F_{kj}(x_k) = \frac{F(x_k + \Delta x_k) - F(x_k - \Delta x_k)}{2\Delta x_{kj}}, \;\; j \in \overline{1,M}. \tag{16}$$

3.5. Create a position vector based on mutation

$$y = (y_1, ..., y_M), \;\; y_j = x_{kj} + \mathrm{sgn}(F_{kj}(x_k))a(n). \tag{17}$$

3.6. If $\forall j \in \overline{1,M} x_j^{\min} \le y_j \le x_j^{\max}$, then $x_k = y$.

3.7. If $m < N^L$, then $m = m + 1$, go to step 3.3.

3.8. If $k \le K$, then $k = k + 1$, go to step 3.2.

4. Observation-jump process.

4.1. Monkey number $k = 1$.

4.2. Create a position vector y based on mutation

$$y = (y_1, ..., y_M), \ \ y_j = x_{kj} + (1 - 2U(0,1))b(n). \tag{18}$$

4.3. If $\left(F(y) < F(x_k)\right) \wedge \left(\forall j \in \overline{1,M} x_j^{\min} \le y_j \le x_j^{\max}\right)$, then $x_k = y$, otherwise go to step 4.2.

4.4. If $k \le K$, then $k = k + 1$, go to step 4.2.

5. Climbing process.

5.1. Monkey number $k = 1$.

5.2. $m = 1$.

5.3. Create a position vector increment Δx_k

$$\Delta x_k = (\Delta x_{k1}, ..., \Delta x_{kM}), \ \ \Delta x_{kj} = round(1 - 2U(0,1))a(n). \tag{19}$$

5.4. Calculate the pseudo-gradient of the cost function

$$F_{kj}(x_k) = \frac{F(x_k + \Delta x_k) - F(x_k - \Delta x_k)}{2\Delta x_{kj}}, \ \ j \in \overline{1,M}. \tag{20}$$

5.5. Create a position vector based on mutation

$$y = (y_1, ..., y_M), \ \ y_j = x_{kj} + \delta(n)\mathrm{sgn}(F_{kj}(x_k))a(n). \tag{21}$$

5.6. If $\forall j \in \overline{1,M} x_j^{\min} \le y_j \le x_j^{\max}$, then $x_k = y$.

5.7. If $m < N^L$, then $m = m + 1$, go to step 5.3.

5.8. If $k \le K$, then $k = k + 1$, go to step 5.2.

6. Somersault process b.

6.1. Monkey number $k = 1$.

6.2. Create a position vector based on the crossover

$$\mu = (\mu_1, ..., \mu_M), \ \ \mu_j = \frac{1}{K}\sum_{l=1}^{K} x_{lj}. \tag{22}$$

6.3. Create a position vector y based on the crossover

$$y = (y_1, ..., y_M), \ \ y_j = x_{kj} + (1 - 2U(0,1))(\mu_j - x_{kj}). \tag{23}$$

6.4. If $\forall j \in \overline{1,M} x_j^{\min} \le y_j \le x_j^{\max}$, then $x_k = y$.

6.5. If $k \le K$, then $k = k + 1$, go to step 6.2.

7. Determine the best monkey based on the cost function

$$k^* = \arg\min_{k \in \overline{1,K}} F(x_k). \tag{24}$$

8. Determine the global best position.

If $F(x_{k^*}) < F(x^*)$, then $x^* = x_{k^*}$.

9. If $n < N$,

then $n = n + 1$, $a(n) = a^{\min} + (a^{\max} - a^{\min})\left(1 - \frac{n}{N}\right)$, $b(n) = b^{\min} + (b^{\max} - b^{\min})\left(1 - \frac{n}{N}\right)$, go to step 3.

The result is x^*.

5. Optimization of Spider Monkey with Dynamic Parameters

Spider Monkey Optimization (SMO) was proposed by Bansal, Sharma, Jadon, and Clerc [28] and is based on the behavior of spider monkeys, which is based on the fission-fusion social structure (FFSS).

The key features of FFSS are: An example of numbered list is as following.

1. Animals based on FFSS are social and live in groups of 40-50 individuals. The FFSS swarm can reduce competition for food among group members by dividing them into subgroups for food search.
2. The female (global leader) usually leads the group and is responsible for finding food sources. If she fails to obtain enough food for the group, she divides the group into smaller subgroups (ranging in size from 3 to 8 members) who forage independently.
3. Subgroups should also be led by a female (local leader) who makes decisions about planning an efficient feeding route for each day.
4. Members of these subgroups communicate within and outside the subgroup depending on the availability of food and to maintain territorial boundaries.

In the developed strategy, the behavior of animals living according to the FFSS principle (such as spider monkeys) is divided into four stages. In the first stage, the group begins to search for food and evaluates the distance to it. In the second stage, based on the distance to the food, group members update their positions and reevaluate the distance to food sources. In the third stage, the local leader updates her best position in the group, and if the position is not updated a certain number of times, all members of this group begin to search for food in different directions. In the fourth stage, the global leader updates her best position, and if it is not updated a certain number of times, she divides the group into smaller subgroups. All four stages are performed continuously until the desired result is achieved. FFSS follows the properties of self-organization and division of labor to achieve intelligent swarm behavior of animals. However, the proposed algorithm differs from the natural behavior of spider monkeys. In this algorithm, the leader's position (local or global) is not constant but depends on the leader's ability to find food. In addition, the algorithm does not model the different communication tactics of spider monkeys. This algorithm consists of six phases: Local Leader Phase, Global Leader Phase, Local Leader Learning Phase, Global Leader Learning Phase, Local Leader Decision-making Phase, and Global Leader Decision-making Phase. Unlike the original method, this method uses the dynamic probability of retaining a solution and the dynamic probability of a random solution.

5.1. Algorithm for optimizing numerical functions

1. Initialization.

1.1. Setting the minimum and maximum probabilities $p^{\min}, p^{\max}$.

1.2. Setting the maximum number of iterations N, the maximum number of groups $G^{\max}$, the maximum number of iterations without updating the position of the local leader N^{LL} (usually $N^{LL} = M \cdot K$), the maximum number of iterations without updating the position of the global leader N^{G} (usually $N^{G} \in [K/2, 2K]$), the population size K ((for integer division of the population into groups we assume $K = 60$), the length of the monkey position vector M, and the minimum and maximum values for the position vector $x_j^{\min}, x_j^{\max}$, $j \in \overline{1, M}$.

1.3. Setting the cost function (objective function) $F(x) \to \min\limits_{x}$,

where x – is the monkey position vector.

1.4. Number of groups $G = 1$.

1.5. Creation of one group.

1.5.1. Monkey number in the group. $k = 1$.

1.5.2. Random creation of a position vector x_{gk}

$$x_{1k} = (x_{1k1}, ..., x_{1kM}), \; x_{1kj} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) U(0,1), \tag{25}$$

where $U(0,1)$ – is a function returning a standard uniformly distributed random number.

1.5.3. If $x_{1k} \notin P_1$, then $P_1 = P_1 \bigcup \{x_{1k}\}$, $k = k + 1$.

1.5.4. If $k \le |P_1|$, then go to step 1.5.2.

1.6. Determine the local leader in the group

$$k^* = \arg\min_{k \in \overline{1, |P_1|}} F(x_{1k}), \;\; x_1^* = x_{1k^*}. \tag{26}$$

1.7. Determine the global leader

$$x^* = x_{1^*}. \tag{27}$$

1.8. Number of iterations without updating the position of the local leader

$$m_1 = 0. \tag{28}$$

1.9. Number of iterations without updating the position of the global leader

$$m = 0. \tag{29}$$

1.10. Probability of preserving the position $p^s(1) = p^{\min}$.

1.11. Probability of a random position $p^r(1) = p^{\max}$.

2. Iteration number $n = 1$.

3. Local leader phase.

3.1. Group number $g = 1$.

3.2. Monkey number в g -group $k = 1$.

3.3. Component number $j = 1$.

3.4. Randomly select the number of the second monkey

$$s = round\left(1 + \left(\left|P_g\right| - 1\right)U(0,1)\right), \tag{30}$$

where $round()$ – is a function that rounds a number to the nearest integer.

3.5. If $s = k$, then go to step 3.4.

3.6. Save the position vector or create a position vector based on the crossover (crossing the position vector of the monkey with the position vectors of the local leader and the other monkey)

$$\lambda = U(0,1),$$

$$\hat{x}_{gkj} = \begin{cases} x_{gkj}, & \lambda < p^s(n) \\ x_{gkj} + U(0,1)(x^*_{gj} - x_{gkj}) + (1 - 2U(0,1))(x_{gsj} - x_{gkj}), & \lambda \geq p^s(n) \end{cases}. \tag{31}$$

3.7. If $j < M$, then $j = j + 1$, go to step 3.4.

3.8. If $k < \left|P_g\right|$, then $k = k + 1$, go to step 3.3.

3.9. If $g < G$, then $g = g + 1$, go to step 3.2.

4. Modify the position of group members.

4.1. Group number $g = 1$.

4.2. Monkey number в g -group $k = 1$.

4.3. If $F(\hat{x}_{gk}) < F(x_{gk})$, then $x_{gk} = \hat{x}_{gk}$.

4.4. If $k < \left|P_g\right|$, then $k = k + 1$, go to step 4.3.

4.5. If $g < G$, then $g = g + 1$, go to step 4.2.

5. Calculate probabilities for group members.

5.1. Group number $g = 1$.

5.2. Monkey number в g -group $k = 1$

$$p_{gk} = 0.1 + 0.9\frac{1/F(x_{gk})}{\max\limits_{s \in 1,\left|P_g\right|} 1/F(x_{gs})}. \tag{32}$$

5.3. If $k < \left|P_g\right|$, then $k = k + 1$, go to step 5.3.

5.4. If $g < G$, then $g = g + 1$, go to step 5.2.

6. Global leader phase.

6.1. Group number $g = 1$.

6.2. $c = 1$.

6.3. If $c \geq \left|P_g\right|$, then go to step 6.13.

6.4. Monkey number в g -group $k = 1$.

6.5. If $U(0,1) \geq p_{gk}$, then go to step 6.11.

6.6. $c = c + 1$.

6.7. Randomly select the component number $j = 1$

$$j = round\left(1 + \left(M - 1\right)U(0,1)\right). \tag{33}$$

6.8. Randomly select the number of the second monkey

$$s = round\left(1 + \left(\left|P_g\right| - 1\right)U(0,1)\right). \tag{34}$$

6.9. If $s = k$, then go to step 6.8.

6.10. Create a position vector based on the crossover

$$\hat{x}_{gkj} = x_{gkj} + U(0,1)(x^*_{gj} - x_{gkj}) + \delta(n)(1 - 2U(0,1))(x_{gsj} - x_{gkj}). \tag{35}$$

6.11. If $k < \left|P_g\right|$, then $k = k + 1$, go to step 6.5.

6.12. Modify the position of group members

$$x_{gk} = \hat{x}_{gk}, \quad k \in \overline{1, \left|P_g\right|}, \quad g \in \overline{1, G}. \tag{36}$$

6.13. If $g < G$, then $g = g + 1$, go to step 6.2.

7. Local leader learning phase.

7.1. Group number $g = 1$.

7.2. Determine the best monkey in the g -group based on the objective function

$$k^* = \arg\min_{k \in 1, \left|P_g\right|} F(x_{gk}). \tag{37}$$

7.3. Determine the local leader in the g -group.

If $F(x_{gk^*}) < F(x^*_g)$, then $x^*_g = x_{gk^*}$, $m_g = m_g + 1$.

7.4. If $g < G$, then $g = g + 1$, go to step 7.2.

8. Global leader training phase.

8.1. Determine the best local leader

$$g^* = \arg\min_{g \in 1, G} F(x^*_g). \tag{38}$$

8.2. Determine the global leader.

If $F(x_{g^*}) < F(x^*)$, then $x^* = x_{g^*}$, otherwise $m = m + 1$.

9. Local leader decision-making phase.

9.1. Group number $g = 1$.

9.2. If $m_g \leq N^{LL}$, then go to step 9.9.

9.3. $m_g = 0$.

9.4. Monkey number в g -group $k = 1$.

9.5. Component number $j = 1$.

9.6. Create a position vector randomly or based on the crossover (crossing the position vector of the monkey with the position vectors of the global and local leaders)

$$\lambda = U(0,1),$$

$$x_{gkj} = \begin{cases} x_j^{\min} + (x_j^{\max} - x_j^{\min})U(0,1), & \lambda < p^r(n) \\ x_{gkj} + U(0,1)(x^*_j - x_{gkj}) + U(0,1)(x_{gkj} - x^*_{gj}), & \lambda \geq p^r(n) \end{cases}. \tag{39}$$

9.7. If $j < M$, then $j = j + 1$, go to step 9.6.

9.8. If $k < \left|P_g\right|$, then $k = k + 1$, go to step 9.5.

9.9. If $g < G$, then $g = g + 1$, go to step 9.2.

10. Global leader decision-making phase.

10.1. If $m \leq N^G$, then go to step 11.

10.2. Number of iterations without updating the position of the global leader $m = 0$.

10.3. Fuse the group into the population

$$P = \bigcup_{g=1}^{G} P_g. \tag{40}$$

10.4. If $G < G^{\max}$, then $G = G + 1$, fission the population P into G groups, go to step 10.6.

10.5. If $G \geq G^{\max}$, then $G = 1$.

10.6. Determine local leaders in groups

$$x_g^* = \arg\min_{k \in \overline{1,|P_g|}} F(x_{gk}), \quad g \in \overline{1,G}. \tag{41}$$

10.7. Number of iterations without updating the positions of local leaders

$$m_g = 0, \quad g \in \overline{1,G}. \tag{42}$$

11. If $n < N$, then $n = n + 1$,

$$p^r(n) = p^{\min} + (p^{\max} - p^{\min})\left(1 - \frac{n}{N}\right), \quad p^r(n) = p^{\min} + (p^{\max} - p^{\min})\left(\frac{n}{N}\right), \tag{43}$$

go to step 3.

The result is x^*.

6. Numerical study of the proposed methods

The numerical study was conducted on the Ackley function

$$f(x) = -a\exp\left(-b\sqrt{\frac{1}{M}\sum_{j=1}^{M} x_j^2}\right) - \exp\left(\frac{1}{M}\sum_{j=1}^{M}\cos(cx_j)\right) + a + \exp(1), \tag{44}$$

where $a = 20, b = 0.2, c = 2\pi$, $x_j^{\min} = -32.768, x_j^{\max} = 32.768$, $M = 30$.

In the study, the population size is $K = 60$ the maximum number of iterations or trees $N = 100$ (for monkey search and monkey algorithm), or (for spider monkey optimization), were used minimum distance between node positions in memory $\varepsilon = \frac{\| x^{\max} - x^{\min} \|}{8}$, a parameter controlling the step size for generating a new solution $\delta = 0.1$, the maximum number of levels for each tree $H = 100$, the maximum number of vertices on H level $N^H = 100$, memory size $L = 10$, minimum and maximum step lengths $a^{\min} = 0.1, a^{\max} = 0.001$ for generating the position vector during the ascent process, minimum and maximum step lengths $b^{\min} = 1, b^{\max} = 10$ for generating the position vector during the observation-jump process, the maximum number of local search iterations $N^L = 1000$, minimum and maximum probabilities $p^{\min} = 0.1, p^{\max} = 0.9$, the maximum number of groups $G^{\max} = 6$, the max number of iterations without updating the position of the local leader $N^{LL} = 1800$, the maximum number of iterations without updating the position of the global leader $N^G = 60$. The results of comparing the proposed methods with classical methods are presented in Table 1.

Table 1

Comparison of proposed metaheuristic methods for parametric identification with existing ones based on mean squared error criterion

Method	Mean squared error	
	of the proposed method with dynamic parameters	of existing method
Monkey search	0.04	0.08
Monkey algorithm	0.03	0.07
Spider monkey optimization	0.02	0.06

An example of a real-world applied problem is neural network training. A multi-layer perceptron with five input neurons, one output neuron, one hidden layer, and 30 weight coefficients was selected as the neural network, performing function approximation.

7. Discussion

The selected parameter values of the proposed parametric identification methods ensure a high probability of mutation and random solution generation in the initial iterations and a low probability of mutation and random solution generation in the final iterations.

Classical methods based on the behavior of monkeys do not take into account the iteration number in the mutation operator and the random solution generation operator, which reduces the accuracy of the solution search (Table 1). The proposed methods allow for the elimination of these shortcomings.

One limitation of the study is that the proposed methods used a population size of 60.

8. Conclusions

In the article, the problem of insufficient efficiency of parameter identification methods for artificial neural network models used in intelligent computer systems is considered. To improve the efficiency of parameter identification methods, metaheuristic methods based on the behavior of monkeys with dynamic parameters were proposed, which improve upon classical methods (monkey search, monkey algorithm, spider monkey optimization). The proposed methods of parameter identification, due to global search at initial iterations and local search at final iterations, allow to increase the accuracy of search and does not require the transformation of the target function. The proposed metaheuristics can expand the scope of application of parameter identification methods based on monkey behavior and contribute to the increase of efficiency of intelligent computer systems. The prospects of further research include the investigation of the proposed methods for a wide range of artificial intelligence tasks.

9. References

[1] Neskorodieva, T., Fedorov, E. Method for automatic analysis of compliance of settlements with suppliers and settlements with customers by neural network model of forecast // Advances in Intelligent Systems and Computing, 2021, 1265 AISC, pp. 156–165. doi:10.1007/978-3-030-58124-4_15

[2] Neskorodieva, T., Fedorov, E. Neural Network Models Ensembles for Generalized Analysis of Audit Data Transformations // Lecture Notes in Networks and Systems, 2022, 344, pp. 263–279.

[3] Aggarwal, C.C. Neural Networks and Deep Learning. Cham, Switzerland: Springer, 2018, 497 p.

[4] S. Kumar, R. Mahapatra. Analytical Analysis of Two-Warehouse Inventory Model Using Particle Swarm Optimization. PCIS 2021, Vol. 2. pp. 215-226. doi:10.1016/j.amc.2014.12.137

[5] Yang, X.-S. Nature-inspired Algorithms and Applied Optimization / X.-S. Yang. – Charm: Springer, 2018. – 330 pp.

[6] Yang, X.-S. Optimization Techniques and Applications with Examples / X.-S. Yang. – Hoboken, New Jersey: Wiley & Sons, 2018. – 364 p.

[7] Nakib, A. Metaheuristics for Medicine and Biology / Nakib A., Talbi El-G. – Berlin: Springer-Verlag, 2017. – 211 p.

[8] Subbotin, S., Oliinyk, A., Levashenko, V., Zaitseva, E. Diagnostic Rule Mining Based on Artificial Immune System for a Case of Uneven Distribution of Classes in Sample. Communications. – Vol.3. – 2016. – P.3-11. doi:10.26552/com.c.2016.3.3-11

[9] Blum, C., Raidl, R. Hybrid Metaheuristics. Powerful Tools for Optimization. – Charm: Springer, 2016. – 157 p.

[10] Bozorg-Haddad, O., Solgi, M., Loaiciga, H. Meta-heuristic and Evolutionary Algorithms for Engineering Optimization. New Jersey: Wiley & Sons, 2017. – 293 p. doi:10.1002/9781119387053

[11] Chopard, B., Tomassini, M. An Introduction to Metaheuristics for Optimization / B. Chopard,. – New York: Springer, 2018. – 230 p.
[12] Radosavljević, J. Metaheuristic Optimization in Power Engineering. – New York: The Institution of Engineering and Technology, 2018. – 536 p. doi:10.1049/PBPO131E
[13] Alba, E., Nakib, A., Siarry, P. Metaheuristics for Dynamic Optimization. – Berlin: Springer-Verlag, 2013. – 398 p.
[14] Nozari, H., Aliahmadi, A., Jafari-eskandari, M., Khaleghi, Gh. An Extended Compact Genetic Algorithm for Milk Run Problem with Time Windows and Inventory Uncertainty // International Journal of Applied Operational Research. - Vol. 5, No. 2. - 2015. - pp. 35-48
[15] Nametala, C.A.L., Faria, W.R., Júnior, B.R.P.. On the performance of the Bayesian optimization algorithm with combined scenarios of search algorithms and scoring metrics // Genetic Programming and Evolvable Machines. - Vol. 23, Issue 2. - 2022. - pp 193–223
[16] Pinto, C. , Runkler, T.A., Sousa, J.M. Wasp Swarm Algorithm for Dynamic MAX-SAT Problems. Proceedings of the 8th international conference on Adaptive and Natural Computing Algorithms, Part I. – 2007. – P. 350-357.
[17] Sharma, A., Sharma, H., Khandelwal, A., Sharma, N.. Designing Controller Parameter of Wind Turbine Emulator Using Artificial Bee Colony Algorithm. Proceedings of Congress on Intelligent Systems 2020, pp. 143-151
[18] Neskorodieva, T., Fedorov, E., Chychuzhko, M., Chychuzhko, V. Metaheuristic Method for Searching Quasi-Optimal Route Based On the Ant Algorithm and Annealing Simulation // Radioelectronic and Computer Systems, 2022, 2022(1), pp. 92–102
[19] Balamurugan, R., Natarajan, A.M., Premalatha, K. Stellar-mass black hole optimization for biclustering microarray gene expression data. Appl Artif Intell. – 2015. – Vol. 29. – P. 353-81. doi:10.1080/08839514.2015.1016391
[20] Mirjalili, S., Mirjalili, S.M., Hatamlou, A. Multi-verse optimizer: a nature-inspired algorithm for global optimization. NCA – 2015. – Vol. 49. – P. 1–19. doi:10.1007/s00521-015-1870-7
[21] Cuevas, E., Echavarria, A., Ramirez-Ortegon, M.A. An optimization algorithm inspired by the states of matter that improves the balance between exploration and exploitation. Appl Intell. – 2014. – Vol. 40. – P. 256–272. doi:10.1007/s10489-013-0458-0
[22] Patel, V.K., Savsani, V.J. Heat transfer search (HTS): A novel optimization algorithm. Inf Sci. – 2015. – Vol.324. – P. 217–246. doi:10.1016/j.ins.2015.06.044
[23] Moein S., Logeswaran, R. KGMO: A swarm optimization algorithm based on thekinetic energy of gas molecules. Inf Sci. – 2014. – Vol. 275. – P.127–144. doi:10.1016/j.ins.2014.02.026
[24] Hatamlou A. Black hole: A new heuristic optimization approach for data clustering. Inf Sci. – 2013. – Vol.222. – P.175–184. doi:10.1016/j.ins.2012.08.023
[25] Yan, G-W., Hao, Z.-J. A novel optimization algorithm based on atmosphere clouds model. Int J Comput Intell – Vol. 12, №1. – 2013 – P. 16.
[26] Shareef, H., Ibrahim, A.A., Mutlag, A.H. Lightning search algorithm. Appl Soft Comput. – 2015. – Vol. 36. – P. 315-33. doi:10.1016/j.asoc.2015.07.028
[27] Kaveh, A., Khayatazad, M. A new meta-heuristic method: ray optimization. Comput Struct. – 2012. – Vol. 112. – P. 283-294. doi:10.1016/j.compstruc.2012.09.003
[28] Kashan, A.H. A New metaheuristic for optimization: optics inspired optimization (OIO). Technical Report, Department of Industrial Engineering, Tarbiat Modares University. – 2013. doi:10.1016/j.cor.2014.10.011