ISSN 0005-1179 (print), ISSN 1608-3032 (online), Automation and Remote Control, 2023, Vol. 84, No. 8, pp. 1005–1016.
© The Author(s), 2023 published by Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, 2023.
Russian Text © The Author(s), 2023, published in Avtomatika i Telemekhanika, 2023, No. 8, pp. 153–168.

Control of Set of System Parameter Values by the Ant Colony Method

I. N. Sinitsyn[*,**,a] **and Yu. P. Titov**[*,**,b]

[*]*Moscow Aviation Institute (National Research University), Moscow, Russia*
[**]*Federal Research Center for Computer Science and Control, Russian Academy of Sciences, Moscow, Russia*
e-mail: [a]*sinitsin@dol.ru,* [b]*kalengul@mail.ru*

Received January 23, 2023
Revised March 21, 2023
Accepted June 9, 2023

Abstract—The paper considers the modification and application of the ant colony method for the problem of directed enumeration of the values of system parameters when performing calculated multiple calculations. Interaction with the user makes it possible to stop the process of exhaustive enumeration of sets of parameter values, and the application of a modification of the ant colony method will allow us to consider rational sets at early iterations. If the user does not terminate the algorithm, then the proposed modifications allow one to enumerate all solutions using the ant colony method. To modify the ant colony method, a new probabilistic formula and various algorithms of the ant colony method are proposed, allowing for each agent to find a new set of parameter values. The optimal algorithm, according to the research results, is the use of repeated endless cyclic search for a new solution. This modification allows you to consider all solutions, and at the same time, find all the optimal solutions among the first 5% of the considered solutions.

Keywords: ant colony method, parametric graph, reordering, computing cluster, hyperparameter optimization

DOI: 10.25728/arcRAS.2023.91.87.001

1. INTRODUCTION

Nowadays, due to the development of computing clusters, many computational and optimization tasks are transferred from human experts to computing machines. For such systems arise the problems of finding rational values of parameters for solving the computational problem, called hyperparameter optimization [1]. Among the algorithms of hyperparameter optimization one can mention the Bayesian optimizer, which allows finding regularities of influence of individual parameter values (and various combinations) on performance criteria on the basis of hypotheses and a posteriori information [2]. For multi-criteria and multi-extreme problems it is often necessary to consider all sets of parameter values, usually by the method of complete search. This paper considers the possibility of using the ant colony method to solve the problem of directed enumeration of sets of hyperparameters before sending them to a computing system. By interacting with the user, it is possible to stop the method before considering all sets of parameter values when a set satisfying the user's conditions is found. Directed search by the ant colony method will allow to consider rational sets of parameter values as early as possible, but in case of user's dissatisfaction with the results, it will be possible to reconsider all sets of values.

The ant colony method was originally developed for solving the traveling salesman problem [3, 4]. Modern research allows to apply the ant colony method to search for continuous optimization. The search methods of CACO (ContinuousAntColonyOptimization) [5], ACOR (Ant Colony Optimization for continuous domain) [6] and CIAC (ContinuousInteractingAntColony) [7, 8] do not involve the use of a graph and have been actively investigated by different researchers, including researchers from Russia [9, 10]. Studies describe the possibility of using the ant colony method for solving problems on graphs: assignment problems with fuzzy execution time [11, 12], problems of finding optimal routes for a group of salesmen [13, 14] and problems of supporting spare parts supply processes [15]. Parametric problems related to finding an optimal set of parameters, classification, dependency detection, etc. are actively researched in the global community [16–19]. For such problems, a special graph structure is created. The presented methods and modifications of the ant colony method are designed to find approximate and rational solutions. Usually all agents (ants) should converge to one solution. The search for new solutions is carried out by the multistart [9].

For a directed search of parameter values, it is necessary not to converge to one solution (set of parameter values) but to consider new solutions sequentially until the user stops the method or considers all possible solutions. In this paper, modifications of the algorithm are proposed to consider all solutions instead of converging to one. The proposed approach allows to solve problems with vector optimality criterion and multimodal target functions without restarting the algorithm. At the same time, the property of optimization algorithms, the fastest finding of optimal solutions, is preserved.

2. MODELS AND METHODS

The ant colony method is based on the probabilistic search for an arc in a graph according to the formula

$$P_{ij,k}(t) = \frac{\tau_{ij}^{\alpha} \times \mu_{ij,t}^{\beta}}{\sum\limits_{z \in J_{i,k}} \left(\tau_{iz}^{\alpha} \times \mu_{iz,t}^{\beta} \right)}. \tag{1}$$

Using (1), the transition probability of an agent from the current vertex i to the vertices from the set $J_{i,k}$ at iteration t is determined. From the computational results, the transition probability to vertex j for the kth agent is determined. The (1) takes the arc length information τ_{ij}^{α} (remoteness) and some weight $\mu_{ij,t}^{\beta}$ (pheromone) into account. The value of τ_{ij}^{α} is fixed and does not depend on the iteration number t. The number of weights $\mu_{ij,t}^{\beta}$ changes between iterations, updating the state of the graph and the external environment for the agents' movement.

To determine the values of the system (solution) parameters, it is suggested to represent the sets of values in the form of a parametric graph. Each specific value of one parameter represents a vertex of the graph. All values of one parameter are combined into layers. From each vertex of one layer there is an arc to each vertex of the next layer. The layers of vertices are arranged in a certain order, which reduces the number of arcs in the graph. An example of a parametric graph is shown in Fig. 1. Similar graphs are found in [13, 15, 18–21].

The weights (pheromone) in such a graph are recorded not on arcs but on vertices. As a result, the arcs are fictitious, and such a parametric graph can be represented as a set of layers (parameters) and a set of vertices (parameter values) [22].

For a parametric graph in the probabilistic formula (1), the value τ_{ij}^{α} can only be set on the basis of a priori information from an expert, but this parameter cannot be defined in general. If a single factor is used in the probability formula, the stagnation of the solution search process increases. The algorithm converges to the first good solution and does not continue the search for the optimal one.

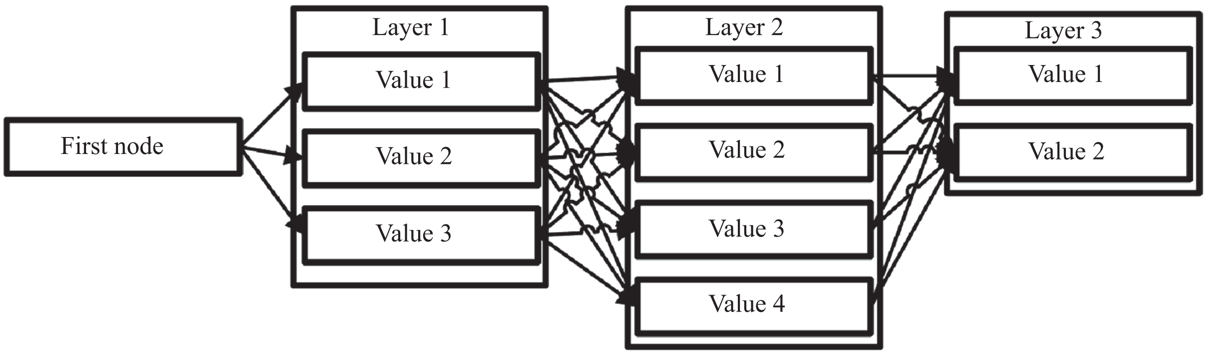

Fig. 1. Diagram of the parametric graph structure.

To solve the stagnation problem, we can "reset" the parametric graph, transfer the state of the graph vertices to the initial state, or modify the probabilistic formula

$$P_{ij,k}(t) = \frac{k1 \times \mu^{\alpha}_{norm\,ij,t} + k2 \times \left(\frac{1}{kol(t)_{ij,t}}\right)^{\beta} + k3 \times \left(\frac{kol(t)_{ij,t}}{MaxKol_{j,t}}\right)^{\gamma}}{\sum\limits_{z \in J_{i,k}} \left(k1 \times \mu^{\alpha}_{norm\,iz,t} + k2 \times \left(\frac{1}{kol(t)_{iz,t}}\right)^{\beta} + k3 \times \left(\frac{kol(t)_{iz,t}}{MaxKol_{z,t}}\right)^{\gamma}\right)}. \tag{2}$$

The formula (2) is a linear convolution (not multiplicative as in (1)) of three summands and weights. The first summand $\mu^{\alpha}_{norm\,ij,t}$ is determined by the number of weights at the ith vertex of the jth parameter (parametric graph layer) at iteration t. To apply this parameter in the weighted sum, it is necessary to use the normalized value. The second summand $(\frac{1}{kol(t)_{ij,t}})^{\beta}$ is determined by the number of visits of agents to the ith vertex of the jth parameter during the running time of the algorithm. This summand increases the probability of visiting a vertex that is rarely present in the solutions and avoids stagnation in the early iterations of the ant colony method. The third summand $(\frac{kol(t)_{ij,t}}{MaxKol_{j,t}})^{\gamma}$ considers the maximum number of possible visits to a vertex: the values of $MaxKol_{j,t}$ for the parameter j at iteration t. Since, in a parametric graph, one vertex (one parameter value) must be selected on each layer, we can calculate the total number of solutions or sets of parameter values. The total number of solutions can be calculated as the product of the number of vertices in each layer. The maximum number of solutions that can contain a particular vertex of a parametric graph is computed as the ratio of the total number of solutions and the number of vertices in a given layer, i.e., for each vertex of layer j, the value $MaxKol_{j,t}$ will be the same. The third summand at later iterations allows to increase the probability of choosing the vertex, for which the majority of solutions are considered. If all possible solutions are considered for a vertex, this vertex can be excluded from the probabilistic search. Additive convolution allows to compensate the values of summands. Vertices with a large number of weights and frequent visits (frequently considered vertices) can be compensated by vertices with a small number of visits (rare vertices) or vertices, for which almost all solutions are considered.

Another specific feature of the algorithm modification is the need to interact with an external computing system. For such modifications, it is necessary to store the state of the system computed for a particular solution. If the ant colony method repeatedly finds a solution, this solution is not sent to the calculator again, and the value of the target function is taken from the hash table [15, 21].

For the considered algorithm, it is important to find a new solution at each iteration. Therefore, if the solution already exists in the hash table, different actions are possible:

1. Using the target function values from the hash table, enter the weights as in the original algorithm.

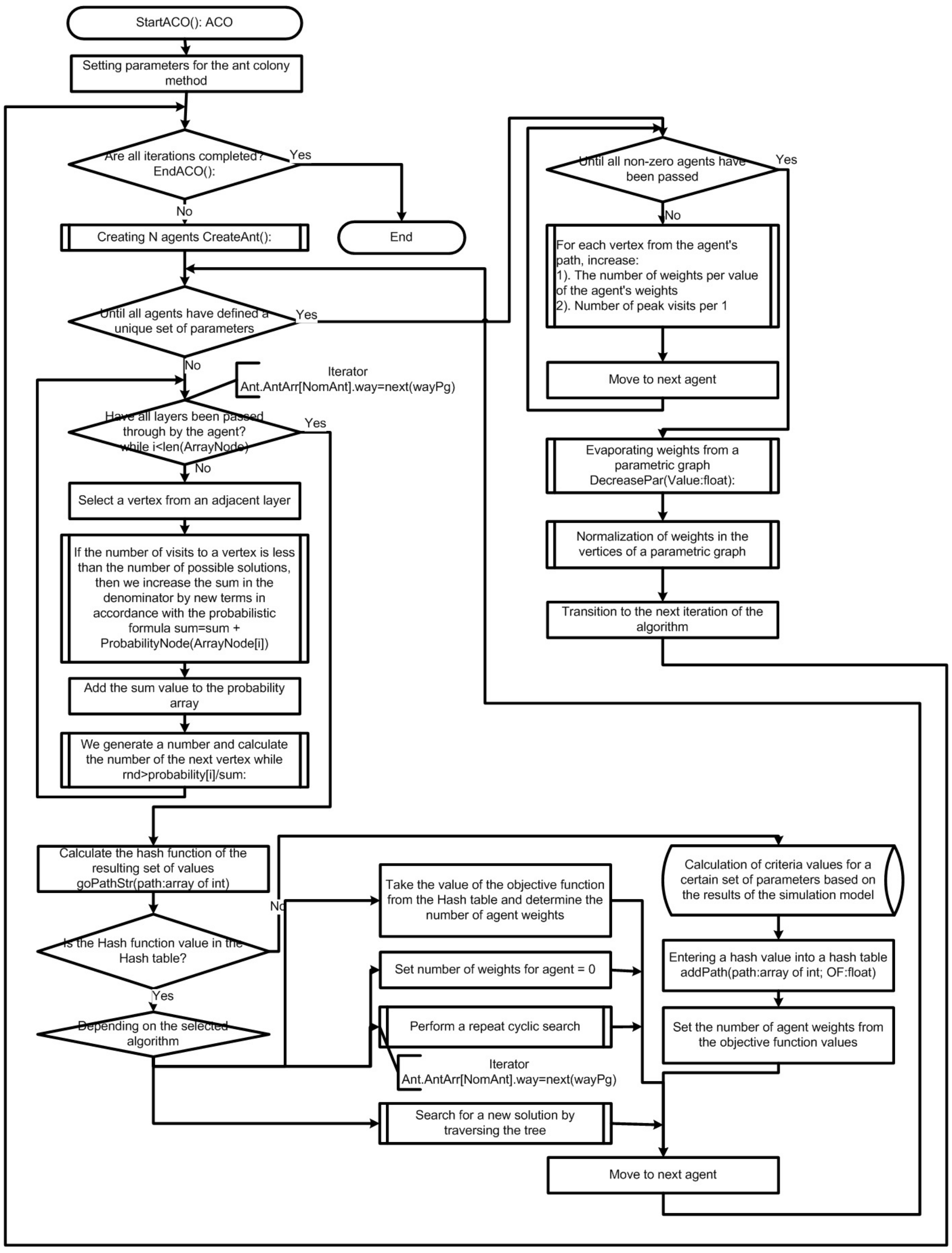
StartACO(): ACO
Setting parameters for the ant colony method
Are all iterations completed? EndACO():
Yes
No
End
Creating N agents CreateAnt():
Until all agents have defined a unique set of parameters
Yes
No
Iterator Ant.AntArr[NomAnt].way=next(wayPg)
Have all layers been passed through by the agent? while i<len(ArrayNode)
Yes
No
Select a vertex from an adjacent layer
If the number of visits to a vertex is less than the number of possible solutions, then we increase the sum in the denominator by new terms in accordance with the probabilistic formula sum=sum + ProbabilityNode(ArrayNode[i])
Add the sum value to the probability array
We generate a number and calculate the number of the next vertex while rnd>probability[i]/sum:
Until all non-zero agents have been passed
Yes
No
For each vertex from the agent's path, increase:
1). The number of weights per value of the agent's weights
2). Number of peak visits per 1
Move to next agent
Evaporating weights from a parametric graph DecreasePar(Value:float):
Normalization of weights in the vertices of a parametric graph
Transition to the next iteration of the algorithm
Calculate the hash function of the resulting set of values goPathStr(path:array of int)
Is the Hash function value in the Hash table?
No
Yes
Depending on the selected algorithm
Take the value of the objective function from the Hash table and determine the number of agent weights
Set number of weights for agent = 0
Perform a repeat cyclic search
Iterator Ant.AntArr[NomAnt].way=next(wayPg)
Search for a new solution by traversing the tree
Calculation of criteria values for a certain set of parameters based on the results of the simulation model
Entering a hash value into a hash table addPath(path:array of int; OF:float)
Set the number of agent weights from the objective function values
Move to next agent

Fig. 2. Schematic diagram of the algorithm of the modified ant colony method.

2. Ignore the agent. The agent does not put weights on the parametric graph.
3. Repeated search for a new solution, not yet considered on the calculator, by the ant colony method with a limited number of iterations. If no new solution is found for the set number of iterations, the agent is ignored.
4. Repeated search for a new solution, not yet considered on the calculator, by the ant colony method with an unlimited number of iterations. The restriction in item no. 3 can solve the stagnation problem.
5. Repeat the search for a new solution by another algorithm. The possibility of traversing the parametric graph as a tree has been considered.

The algorithm flowchart of the proposed modification of the ant colony method is shown in Fig. 2.

3. EXPERIMENT

The objectives of directed enumeration of parameter values are:

—enumeration of all sets of parameter values without any exception;

—the fastest possible acquisition of the optimal set of parameter values.

It can be noticed that both problems contradict each other, because among all sets of parameter values there will always be the best one. But since the system interacts with the calculator and the user in real time, there is a possibility of stopping the program by the user if a satisfactory solution is found. It should also be noted that it is possible to use the vector criterion, which transfers the problem into the area of decision support and multi-criteria optimization [20].

The effectiveness of modifications of the ant colony method is determined by two main evaluations:

—ability of the algorithm to consider all solutions. It is determined by the value of the criterion: "estimate of the probability of finding a new solution by the agent." This estimate is calculated by the ratio of the number of solutions found during the algorithm's running time to the total number of solutions considered;

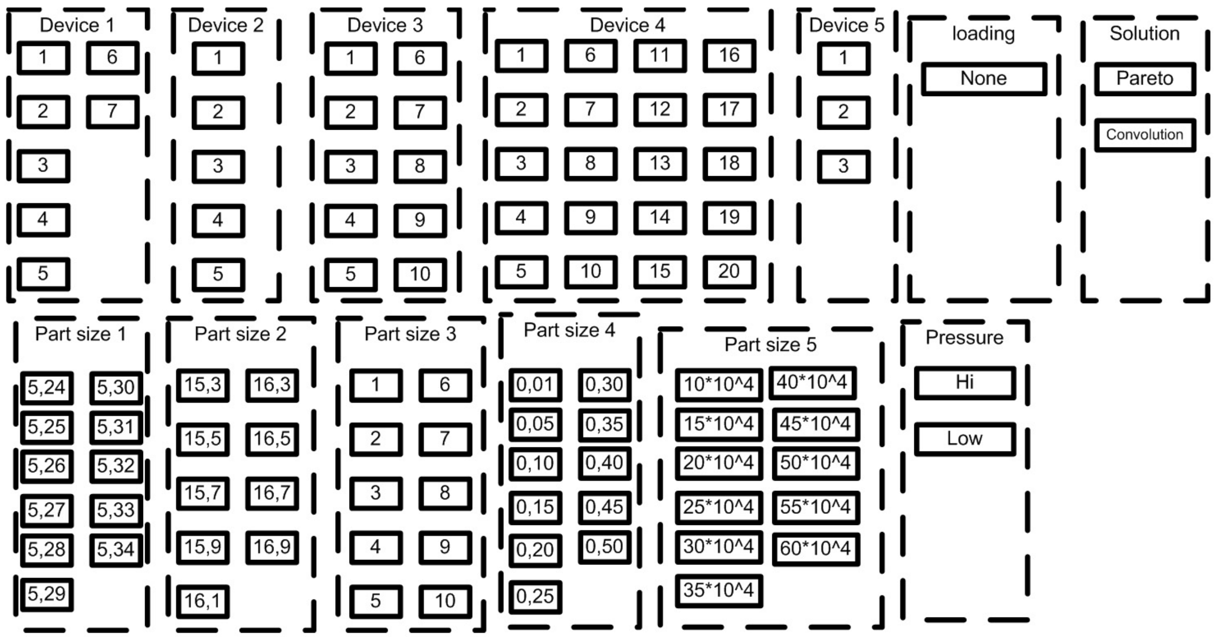

Fig. 3. Diagram of a parametric graph of high dimensionality.

—speed of finding the optimal solution. It is determined by the value of the criterion: "estimate of the expected value of the number of iterations at which this solution was found."

The experiments were performed on the ACO Cluster software written using the Python language. Most of the benchmarks taken from [22, 23] and test graphs of high dimensionality [21] were considered as test data. The structure of the high-dimensional graph is shown in Fig. 3. The presented high-dimensional graph contains more than 10^9 solutions for 13 parameters given in discrete and qualitative scales. With 25 agents per iteration and 20 000 iterations, only 0.05% of the total number of solutions can be considered at most.

To analyze the performance of the algorithm in finding the last (not yet considered) solutions, investigation of the Carrom table function are given in the results:

$$f(x) = -\frac{\left(\cos(x_1)\cos(x_2)e^{|100-(x_1^2+x_2^2)^{0.5}|}\right)^2}{30}. \tag{3}$$

The parametric graph for the function (3) contains two layers: for parameters x_1 and x_2. Each layer of the parametric graph has 201 vertices defining specific parameter values in the range $[-10, 10]$ with precision of 0.1. It takes 1936 iterations of the ant colony method to consider all solutions by 25 agents, assuming that each agent finds a new solution.

4. RESULTS

In this paper, the following modifications of the ant colony algorithm are investigated:

—ACOCC (ACO Cluster Classic)—Classic ant colony method. In (2), the coefficients $k2$ and $k3$ are equal to zero. In this case, the additive convolution formula 2 becomes the multiplicative convolution of the formula (1) with parameter $\tau_{ij}^{\alpha} = 1$. If the solution already considered by the algorithm is found, the value of the target function is determined from the hash table.

—ACOCN (ACO Cluster New)—Similar to the classic ant colony method, but in (2) the coefficients $k2$ and $k3$ are equal to one. The transition probability is determined by additive convolution.

—ACOCNI (ACO Cluster New Ignor)—In (2), the coefficients $k2$ and $k3$ are equal to one. If a solution already written to the hash table is found, this agent is ignored and it does not put weights on the vertices of the parametric graph.

—ACOCCy3 (ACO Cluster Cycle3)—In (2), the coefficients $k2$ and $k3$ are equal to one. If a solution already written to the hash table is found, the solution is searched again using the ant colony method. The search for a new solution is performed cyclically. A limit on the number of iterations of the repeated search is set to 3. If no new solution is found within the set number of iterations, the agent is ignored.

—ACOCCyI (ACO Cluster Cycle Infinity)—In (2), the coefficients $k2$ and $k3$ are equal to one. If a solution already written to the hash table is found, the solution is searched again using the ant colony method. The search for a new solution is performed cyclically with no limit on the number of iterations.

—ACOCT (ACO Cluster Tree)—In (2), the coefficients $k2$ and $k3$ are equal to one. If a solution already written to the hash table is found, the new solution is searched again with a different algorithm. The traversal of the parametric graph as a tree is considered.

Estimates of the results of the modifications for a parametric graph of high dimensionality are given in Tables 1–4. When applying the ACOCN algorithm, the number of solutions required to find the best set of parameter values is minimal, and weakly increases with the number of iterations (3rd column of Table 4). Only the ACOCC algorithm finds a solution faster, but the probability of finding an optimal solution by the ACOCC algorithm is less than 0.1 (2nd column of Table 3). The

Table 1. Estimation of expected value of time of solution search by one agent (in seconds)

Number of iterations	ACOCC	ACOCN	ACOCNI	ACOCCy3	ACOCCyI	ACOCT
2500	1.404×10^{-4}	1.547×10^{-4}	1.562×10^{-4}	1.627×10^{-4}	1.619×10^{-4}	1.887×10^{-4}
5000	1.381×10^{-4}	1.517×10^{-4}	1.560×10^{-4}	1.648×10^{-4}	1.636×10^{-4}	2.745×10^{-4}
7500	1.388×10^{-4}	1.505×10^{-4}	1.567×10^{-4}	1.665×10^{-4}	1.647×10^{-4}	4.158×10^{-4}
10 000	1.391×10^{-4}	1.501×10^{-4}	1.578×10^{-4}	1.690×10^{-4}	1.654×10^{-4}	5.645×10^{-4}
12 500	1.370×10^{-4}	1.547×10^{-4}	1.562×10^{-4}	1.706×10^{-4}	1.657×10^{-4}	6.980×10^{-4}
15 000	1.364×10^{-4}	1.526×10^{-4}	1.569×10^{-4}	1.700×10^{-4}	1.650×10^{-4}	8.593×10^{-4}
17 500	1.328×10^{-4}	1.472×10^{-4}	1.585×10^{-4}	1.695×10^{-4}	1.655×10^{-4}	9.776×10^{-4}
20 000	1.325×10^{-4}	1.469×10^{-4}	1.582×10^{-4}	1.741×10^{-4}	1.688×10^{-4}	10.549×10^{-4}

Table 2. Estimation of the probability of finding a new solution by a single agent per iteration of the ant colony method

Number of iterations	ACOCC	ACOCN	ACOCNI	ACOCCy3	ACOCCyI	ACOCT
2500	0.248	0.618	0.966	1.000	1.000	1.000
5000	0.122	0.381	0.951	1.000	1.000	1.000
7500	0.082	0.282	0.942	1.000	1.000	1.000
10 000	0.063	0.223	0.935	1.000	1.000	1.000
12 500	0.049	0.184	0.929	1.000	1.000	1.000
15 000	0.041	0.158	0.925	1.000	1.000	1.000
17 500	0.035	0.136	0.921	1.000	1.000	1.000
20 000	0.030	0.125	0.918	1.000	1.000	1.000

Table 3. Estimation of the probability of finding a new solution by a single agent per iteration of the ant colony method

Number of iterations	ACOCC	ACOCN	ACOCNI	ACOCCy3	ACOCCyI	ACOCT
2500	0.03	0.31	0.21	0.27	0.24	0.13
5000	0.03	0.49	0.26	0.26	0.33	0.25
7500	0.03	0.50	0.35	0.30	0.32	0.24
10 000	0.02	0.60	0.22	0.32	0.31	0.33
12 500	0.03	0.78	0.32	0.37	0.42	0.27
15 000	0.02	0.71	0.31	0.44	0.37	0.31
17 500	0.03	0.74	0.28	0.39	0.44	0.26
20 000	0.03	0.72	0.41	0.46	0.43	0.39

Table 4. Estimation of expected value of the solution number (in %), at which the optimal parameter values were found

Number of iterations	ACOCC	ACOCN	ACOCNI	ACOCCy3	ACOCCyI	ACOCT
2500	0.801×10^{-6}	2.956×10^{-6}	3.404×10^{-6}	2.970×10^{-6}	3.299×10^{-6}	2.863×10^{-6}
5000	1.122×10^{-6}	3.191×10^{-6}	4.501×10^{-6}	4.415×10^{-6}	5.648×10^{-6}	5.030×10^{-6}
7500	1.014×10^{-6}	3.378×10^{-6}	5.701×10^{-6}	6.137×10^{-6}	5.830×10^{-6}	6.838×10^{-6}
10 000	1.142×10^{-6}	3.667×10^{-6}	5.693×10^{-6}	5.337×10^{-6}	5.910×10^{-6}	7.704×10^{-6}
12 500	0.829×10^{-6}	3.770×10^{-6}	6.779×10^{-6}	6.320×10^{-6}	8.108×10^{-6}	5.269×10^{-6}
15 000	0.740×10^{-6}	3.741×10^{-6}	8.240×10^{-6}	10.174×10^{-6}	8.794×10^{-6}	7.834×10^{-6}
17 500	1.393×10^{-6}	3.845×10^{-6}	8.175×10^{-6}	9.221×10^{-6}	11.678×10^{-6}	15.820×10^{-6}
20 000	1.119×10^{-6}	3.936×10^{-6}	9.864×10^{-6}	11.344×10^{-6}	10.513×10^{-6}	23.592×10^{-6}

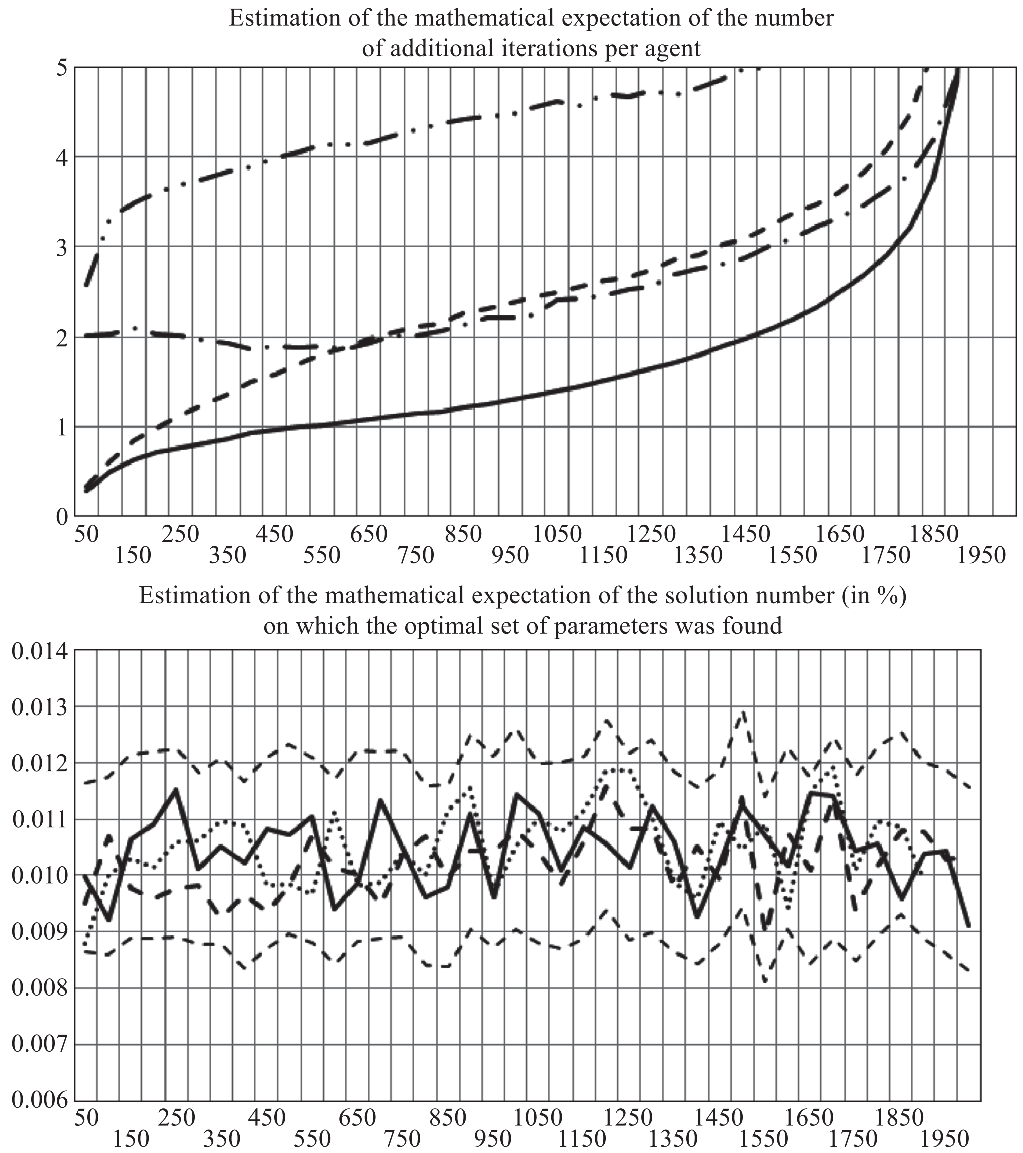

Fig. 4. Dependence of the efficiency of the algorithm on the number of iterations of the ant colony method for the parametric Carrom table function graph.

ACOCC algorithm stagnates at the nearest rational solution. The algorithms that use repeated search for zero agents (ACOCCy3, ACOCCyI and ACOCT) find a new solution for each agent (columns 5–7 of Table 2). For 25 agents, for 2×10^4 iterations, the presented algorithms will consider 5×10^5 sets of parameter values. The time taken for the agent to find a solution does not depend on the number of iterations of the algorithm for most algorithms (Table 1). As a result, it is possible to predict the required number of iterations to find all solutions. The exception is the ACOCT algorithm, which requires more time for one agent to search for a path as the number of iterations increases. The algorithms ACOCC, ACOCN, ACOCNI, ACOCCy3 and ACOCCyI can be ranked in ascending order based on the agent's solution search time. The time of search for a solution by an agent in algorithms ACOCCy3 and ACOCCyI is close, because the number of additional iterations is most often less than the limit of 3 iterations.

It should be noted that the cyclic search actually increases the number of iterations, since it works according to the rules of the ant colony method. If, among ten agents in an iteration, one

Table 5. Estimation of expected value of number of additional iterations by ACOCCyI modification

Number of iterations	1900	1910	1920	1930	1940	1950
$k2$=0; $k3$=0;	8.379	8.712	9.582	10.487	23.004	25.153
$k2$=0; $k3$=0; *ignor*;	8.399	9.028	9.880	10.719	19.658	18.996
$k2$=0; $k3$=1;	5.035	5.248	5.735	6.536	13.408	12.861
$k2$=0; $k3$=1; *ignor*;	5.054	5.291	5.711	6.558	10.658	10.299
$k2$=1; $k3$=0;	7.049	7.780	8.990	11.493	44.962	42.247
$k2$=1; $k3$=0; *ignor*;	7.113	7.744	8.698	10.263	13.658	13.533
$k2$=1; $k3$=1;	4.864	5.312	5.990	7.383	19.585	19.828
$k2$=1; $k3$=1; *ignor*;	4.874	5.314	5.977	7.268	11.172	11.292

agent did not find a new solution and needed ten additional iterations, then twenty iterations were actually performed. Unlike twenty iterations of the original algorithm, there is no updating of weights on the parametric graph for these iterations and the necessary number of iterations can be controlled. Since modifications of ACOCCy3 and ACOCCyI showed good convergence to the optimal solution and each agent in this algorithm finds a new solution (Table 2), let us investigate the possibility of finding all solutions in a parametric graph. For the purpose of the study, we will use a graph of low dimensionality (40 401 solutions) with a target Carrom table function (3). The results of applying the ACOCCyI modification are shown in Fig. 4. Unlike ACOCCyI, ACOCNI modification has considered only 35% of the solutions for 2000 iterations. During the investigation of the ACOCCyI modification, the values of the coefficients $k2$ and $k3$ in (2) were modified.

The long dashed lines with dots in Fig. 4 define the modification that has $k2 = 0$ (at $k3 = 0$ the point is a double point, at $k3 = 1$ the point is a single point). At $k2 = 0$ the optimal solution with probability 1 is determined only after 1000 iterations (more than 60% of the considered solutions). More efficient are the modifications in which $k2 = 1$ (at $k3 = 0$ the line is dashed, at $k3 = 1$ the line is solid). For any number of iterations, the optimal solution is found after examining from 0.009% ($40\,401 \times 0.009 = 363$) solutions to 0.012% (484 solutions)—a confidence interval with a confidence level of 0.99 (bottom plot of Fig. 4). Finding the 0.99% optimal solution requires 2 times less than the considered solutions, sets of parameter values. The ACOCNI modification (line of dots) requires a similar number of considered solutions to find the optimal one.

The top graph of Fig. 4 shows the number of additional iterations required per agent. On average, an agent requires less than 5 additional iterations when considering most possible solutions. As a result, it is inefficient to set a constraint as in the ACOCCy3 modification. The minimum number of additional iterations required to find a new, not yet considered, set of parameter values guarantees minimal delays in the running time of the algorithm.

The inefficient behavior of the modification occurs at the last 36 iterations, at which it is necessary to find the last 900 remaining unexamined solutions (Table 5). The even rows of the table labeled *ignor* are the results of the ACOCCyI modification with the condition of ignoring the vertices for which all solutions are considered. The modification does not consider vertices with $kol(t)_{ij,t} = MaxKol_{j,t}$ in the probability formula (2).

The results presented in Table 5 prove the efficiency of using the third summand in the formula (2). When $k3 = 1$, the number of additional iterations required is significantly smaller than when $k3 = 0$. The smaller number of additional iterations corresponds to a shorter search time for the remaining unexamined sets of parameter values. Ignoring the vertices, for which all possible solutions are considered (lines labeled "ignor"), shows high efficiency.

Both test parametric graphs have several optimal parameter values (equal to the value of the target function). Figure 5 shows the Carrom table function graph and the solution number estimation graph, where the specific values of the parameters x_1 and x_2 were evaluated for 3000 runs of

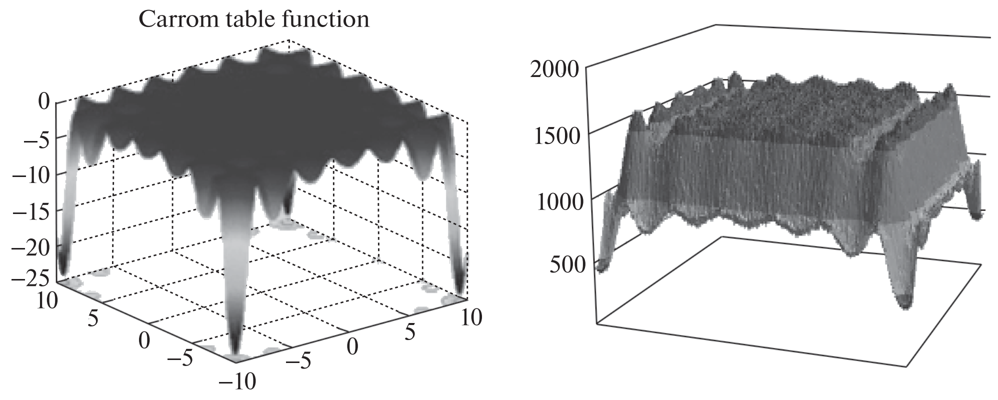

Fig. 5. Function graph and solution number estimates for Carrom table function.

the ACOCCyI modification of the ant colony method. With a large number of runs, the algorithm finds all 4 optima at the earliest iterations. The graph of iteration number estimation visually follows the graph of the function, i.e., the optimal and rational solutions proposed by the modification of the ant colony method are statistically determined at the initial runs.

5. CONCLUSION

In this paper, the problem of directed enumeration of sets of parameter values was considered. To solve the problem, a parametric graph is used, in which the set of parameters is represented as a set of vertices united in groups (layers). This graph has no arcs, and it is necessary to determine the order of the layers, which can lead to differences in the effectiveness of the proposed modifications. But the advantage of this type of parametric graph is the simplicity of its creation for the user. For the ant colony method, we considered the problem of enumerating all values of the system parameters. Optimization properties of the method allow to consider the optimal set of parameter values as early as possible. Modifications are proposed to simplify and control the search for new sets of parameter values using the ant colony method. The use of an interface for interaction with the user is assumed to provide the possibility of stopping the method when a set of system parameter values satisfying the user is found. If such a set is not found, the ant colony method will consider all combinations of discrete parameter values. This approach will also allow to solve multi-extremal and multi-criteria problems.

The considered ant colony method is a rather powerful tool for solving the given problem due to the probabilistic formula for selecting the next vertex. A simple modification of the probabilistic formula is proposed, which allows to significantly improve the performance of the algorithm. This modification defines the probability of choosing the next vertex as an additive convolution of three components: the weights in the vertex, the number of visits to the vertex by agents, and the number of remaining solutions containing this vertex.

In this paper, modifications are proposed that allow choosing different algorithms of behavior when the agent obtains the already considered solution. The verification of the found solution is performed using a hash table. The repeated cyclic search has shown good convergence to the optimal solution and the best performance when searching for the last sets of parameter values on different parametric graphs. The repeated cyclic search performs worse than the original algorithm with the new probabilistic formula.

Consideration of all possible combinations of values of system parameters is a specific task, since it is possible to search all variants by a complete search and, as a result, to find an optimal solution. Most of the algorithms of search for system variants are aimed at finding a rational solution by convergence to it, and such algorithms do not consider "bad" solutions. Despite the probabilistic convergence of the ant colony method to a single solution, the paper proves that it is possible for the algorithm to consider all solutions, even all suboptimal solutions, under any distribution of the probability formula. For multimodal functions, the multistart procedure, which introduces additional stochasticity into the algorithm, can be abandoned.

The disadvantages of the ant colony method include the presence of a large number of free parameters [9]. The parameters related to the "classical" ant colony method (number of agents per iteration, weight evaporation factor, and addition factor) are discussed in detail in [21]. When investigating the asynchronous behavior of the ant colony method in interaction with an asynchronous computational cluster, it is assumed to set the number of agents equal to the number of threads performing computations on the cluster. The lack of convergence of the algorithm to a single solution does not require the setting of boundary conditions and multistart parameters. But the presence of coefficients in the new probabilistic formula requires additional research. Various discrete values of weight and degree coefficients in formula (2) have been investigated. The optimal values are those that are equal to 1. It should be noted that the situation when the weight coefficients take real values and sum up to 1 requires further research. The dynamic change of the coefficient values also requires further research, since the coefficient k_1 is effective in the early iterations to quickly find the optimal set of parameter values, and the coefficient k_3—to find the remaining solutions in the last iterations.

Further development is expected in the following directions.

1. The proposed alternative algorithm for finding a new solution by traversing the tree has shown low efficiency and requires improvement.
2. The value of individual parameter layers of a parametric graph has not been considered. This study will allow to allocate the most and the least significant parameters of the technical system.
3. The probabilistic formula for the agent's choice of the next vertex is a very powerful tool, and it can be modified based on the value of individual layers.
4. Further research on the structure of the parametric graph, the partitioning of the layer into sub-layers, layer permutations, and methods of graph formation is needed.
5. Application of the proposed method for solving problems with vector criterion. Research of modifications for fast consideration on cluster of all solutions from the Pareto set.
6. When the ant colony method works together with an asynchronous calculator, it is assumed to obtain the values of the target function asynchronously with respect to the operation of the ant colony method. For this modification, it is proposed that each agent is computed in its own thread, and as a result, the asynchronous operation of the ant colony method is considered. The addition and evaporation of weights from the parametric graph can be performed as separate threads.

REFERENCES

1. Feurer, M., Hutter, F., and Vanschoren, J., Hyperparameter Optimization, in *The Springer Series on Challenges in Machine Learning*, Cham: Springer, 2019. https://doi.org/10.1007/978-3-030-05318-5_1
2. Koehrsen, W., *A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning*, 2018. https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f
3. Colorni, A., Dorigo, M., and Maniezzo, V., Distributed Optimization by Ant Colonies, in *Proc. First Eur. Conf. on Artific. Life*, Paris: Elsevier Publishing, 1992, pp. 134–142.
4. Dorigo, M. and Stützle, T., *Ant Colony Optimization*, Cambridge: MIT Press, 2004.
5. Socha, K. and Dorigo, M., Ant Colony Optimization for Continuous Domains, *Eur. J. Oper. Res.*, 2008, vol. 185, no. 3, pp. 1155–1173. https://doi.org/10.1016/j.ejor.2006.06.046

6. Mohamad, M., Tokhi, M., and Omar, O.M., Continuous Ant Colony Optimization for Active Vibration Control of Flexible Beam Structures, *IEEE International Conf. on Mechatronics (ICM)*, 2011, no. 4, pp. 803–808.
7. Karpenko, A.P. and Chernobrivchenko, K.A., Efficiency of Optimization by a Continuously Interacting Ant Colony Method (CIAC), *Nauka i obrazovanie: scientific edition of Bauman Moscow State Technical University*, 2011, no. 2. https://doi.org/10.7463/0211.0165551
8. Karpenko, A.P. and Chernobrivchenko, K.A., Multimemetic Modification of Hybrid Ant Algorithm for Continuous Optimization HCIAC, *Nauka i obrazovanie: scientific edition of Bauman Moscow State Technical University*, 2012, no. 9. https://doi.org/10.7463/0912.0470529
9. Karpenko, A.P., Modern Search Engine Optimization Algorithms. Algorithms Inspired by Nature, *Bauman Moscow State Technical University publishing house*, Moscow, 2017, 2nd ed.
10. Simon, D., *Algoritmy evolyutsionnoi optimizatsii: prakticheskoe rukovodstvo* (Evolutionary Optimization Algorithms: A Practical Guide), Moscow: DMK Press, 2020.
11. Sudakov, V.A. and Titov, Y.P., Modified Method of Ant Colonies Application in Search for Rational Assignment of Employees to Tasks, in *Proceedings of 4th Computational Methods in Systems and Software 2020*, vol. 2, Vsetin: Springer Nature, 2020. P. 342–348. DOI 10.1007/978-3-030-63319-6_30
12. Khakhulin, G.F. and Titov, Yu.P., Decision Support System for the Supply of Military Aircraft Spare Parts, *Izv. of Samara scientific center of RAS*, 2014, vol. 16, nos. 1–5, pp. 1619–1623.
13. Sinitsyn, I.N. and Titov, Yu.P., *Razvitie stohasticheskikh algoritmov murav'inoi organizatsii* (Development of Stochastic Algorithms for Ant Organization), in *Bionika—60 let. Results and Prospects. Collection of Articles of the First International Scientific and Practical Conference*, Karpenko, A.P., Ed., Dec. 17–19, 2021, Moscow, 2022, pp. 210–220. https://doi.org/10.53677/9785919160496_210_220
14. Titov, Y.P., Modifications of the Ant Colony Method for Aviation Routing, *Autom. Remote Control*, 2015, vol. 76, no. 3, pp. 458–471. https://doi.org/10.1134/S0005117915030091
15. Sudakov, V.A., Bat'kovskii, A.M., and Titov, Yu.P., Speedup Algorithms for Modifying Ant Colony Method for Finding Rational Assignment of Employees to Tasks with Uncertain Completion Times, *Sovremennye informatsionnye tekhnologii i IT-obrazovanie*, 2020, vol. 16, no. 2, pp. 338–350. https://doi.org/10.25559/SITITO.16.202002.338-350
16. Parpinelli, R., Lopes, H., and Freitas, A., Data Mining with an Ant Colony Optimization Algorithm, *IEEE Trans. Evol. Comput.*, 2002, vol. 6, no. 4, pp. 321–332.
17. Junior, I.C., Data Mining with Ant Colony Algorithms, *ICIC. LNCS*, 2013, vol. 7996, pp. 30–38.
18. Martens, D., De Backer, M., Haesen, R., and Vanthienen, J., Classification with Ant Colony Optimization, *IEEE Trans. Evol. Comput.*, 2007, vol. 11, no. 5, pp. 651–665.
19. Pasia, J.M., Hartl, R.F., and Doerner, K.F., Solving a Bi-objective Flowshop Scheduling Problem by Pareto-Ant Colony Optimization, *ANTS*, 2006, pp. 294–305.
20. Titov, Yu.P., Experience in Modeling Supply Planning Using Modifications of the Ant Colony Method in High Availability Systems, *Sistemy vysokoi dostupnosti*, 2018, vol. 14, no. 1, pp. 27–42.
21. Sinitsyn, I.N. and Titov, Yu.P., Optimization of Hyperparameter Ordering of Computational Cluster by Ant Colony Method, *Sistemy vysokoi dostupnosti*, 2022, vol. 18, no. 3, pp. 23–37. https://doi.org/10.18127/j20729472-202203-02
22. Mishra Sudhanshu, K., *Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method*, University Library of Munich, Germany, MPRA Paper, 2006. https://doi.org/10.2139/ssrn.926132
23. Layeb Abdesslem, *New Hard Benchmark Functions for Global Optimization*, 2022. https://doi.org/10.48550/arXiv.2202.04606

This paper was recommended for publication by O.P. Kuznetsov, a member of the Editorial Board