SoftwareX 19 (2022) 101192

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

CIOA: Circle-Inspired Optimization Algorithm, an algorithm for engineering optimization

Otávio Augusto Peter de Souza [a,*], Letícia Fleck Fadel Miguel [b]

[a] Postgraduate Program in Civil Engineering (PPGEC), Federal University of Rio Grande do Sul (UFRGS), Av. Osvaldo Aranha, 99, 90035-190, Porto Alegre/Rio Grande do Sul, Brazil

[b] Department of Mechanical Engineering (DEMEC), Postgraduate Program in Mechanical Engineering (PROMEC), Postgraduate Program in Civil Engineering (PPGEC), Federal University of Rio Grande do Sul (UFRGS), Av. Sarmento Leite, 425, 90050-170, Porto Alegre/Rio Grande do Sul, Brazil

A R T I C L E I N F O

Article history:
Received 20 May 2022
Received in revised form 15 July 2022
Accepted 4 August 2022

Keywords:
Metaheuristic algorithms
Real-world problems
Structural optimization
Circle-Inspired Optimization Algorithm

A B S T R A C T

This paper presents a new, robust and very efficient metaheuristic optimization algorithm, called Circle Inspired Optimization Algorithm (CIOA), for solving constrained and unconstrained engineering optimization problems. The inspiration for the proposed algorithm consists of well-known formulations of the trigonometric circle. CIOA is compared with five other very famous algorithms in ten benchmark function optimization problems, five real-world engineering constrained optimization problems, and also four structural optimization problems for plane and spatial trusses subjected to multiple and different types of constraints. The results obtained demonstrate that the proposed algorithm is more efficient than other famous algorithms, contributing to the accurate and fast solution of complex optimization problems.

Code metadata

Current code version	v01
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-22-00108
Permanent link to reproducible capsule	-
Legal code license	BSD-3-Clause
Code versioning system used	-
Software code languages, tools and services used	MATLAB
Compilation requirements, operating environments and dependencies	MATLAB
If available, link to developer documentation/manual	-
Support email for questions	otavio.souza@ufrgs.br or letffm@ufrgs.br

1. Motivation and significance

The classic optimization algorithms are generally deterministic and gradient-based. These algorithms are very efficient in the optimization of unimodal functions and without discontinuities. However, most engineering optimization problems are non-linear, involve complex multimodal functions and have numerous restrictions that need to be addressed simultaneously. Metaheuristic algorithms are efficient in solving these problems because they are stochastic algorithms in which the optimization process occurs through an exchange between diversification and intensification [1].

Numerous metaheuristic algorithms have been developed in recent decades. Among the main ones are Simulated Annealing (SA) [2], Particle Swarm Optimization (PSO) [3], Differential Evolution (DE) [4], Harmony Search (HS) [5], Artificial Bee Colony (ABC) [6,7], Ant Colony Optimization (ACO) [8], Firefly Algorithm (FA) [9,10], Gravitational Search Algorithm (GSA) [11], Cuckoo Search (CS) [12], Bat Algorithm (BA) [13], Flower Pollination Algorithm (FPA) [14], Krill Herd (KH) [15], Backtracking Search Optimization Algorithm (BSA) [16], Grey Wolf Optimizer (GWO) [17], Search Group Algorithm (SGA) [18], Whale Optimization Algorithm (WOA) [19], Spotted Hyena Optimizer (SHO) [20], Emperor Penguin Optimizer (EPO) [21], Butterfly Optimization Algorithm (BOA) [22], Sooty Tern Optimization Algorithm (STOA) [23], Seagull Optimization Algorithm (SOA) [24], Multi-Operator Differential Evolution (EnMODE) [25], Self-Adaptive Spherical Search (SASS) [26], Tunicate Swarm Algorithm (TSA) [27], Multi Leader Optimizer (MLO) [28], Darts Game Optimizer (DGO) [29], Spring Search Algorithm (SSA) [30] and Rat Swarm Optimizer (RSO) [31]. Binary versions and hybrid versions of existing algorithms have

* Corresponding author.
E-mail addresses: otavio.souza@ufrgs.br (Otávio Augusto Peter de Souza), letffm@ufrgs.br (Letícia Fleck Fadel Miguel).

https://doi.org/10.1016/j.softx.2022.101192

also been released recently, including: BOSA [32], BEPO [33], ESA [34], HGOANM [35], HTSSA [36], CLFD [37] and EOBL-GOA [38].

The use of metaheuristic algorithms is recurrent in many areas of knowledge, such as in research related to safety, health, AI, Multicore Systems, among others [39–44]. In Engineering, several authors have used metaheuristics to solve complex optimization problems that involve numerous constraints that need to be considered simultaneously [45–61]. Since no metaheuristic algorithm can be considered the best for all possible optimization problems, the development of algorithms that accurately and quickly solve specific problems in a given area becomes relevant [62].

Thus, this paper presents the proposal and formulation of a new, robust and very efficient metaheuristic optimization algorithm called the Circle-Inspired Optimization Algorithm (CIOA) to optimize engineering problems. The CIOA is formulated through some well-known properties of the trigonometric circle. The algorithm's search agents describe arc trajectories and are governed by two specific parameters: a user-defined angle and a radius of the circumference that varies with each iteration depending on the evaluation of each search agent. The CIOA's robustness and efficiency are evaluated in ten benchmark functions well known in the literature where the CIOA's performance is compared with five other famous algorithms. In addition, the CIOA is applied to five real-world optimization problems provided for the 'CEC2020 One Goal Restricted Optimization Competition in the Real World' [63] and also four Truss optimization problems subject to multiple constraints. Thus, the main contribution of this paper is to propose and validate a new and effective optimization tool for engineering problems.

2. Software description

This section presents a description of the developed algorithm: CIOA. Initially, an overview of the code's architecture is presented, then details of its implementation and, finally, aspects of the algorithm's functionality.

2.1. Software architecture

The CIOA is an optimization algorithm programmed in MATLAB, composed of two files: Main_Program_CIOA.m and Objective_Function_CIOA.m

Main_Program_CIOA.m is the main file, in which the algorithm formulation is programmed. It is in this file that the user must adjust the algorithm parameters and inform details about the optimization problem. The Objective_Function_CIOA.m is a function file in which the user must implement the objective function he wants to optimize. In this file, inequality and equality constraints (if any) and the penalty adopted will also be programmed.

2.2. Software implementation

In this section, a brief description of the formulation and implementation of the CIOA is given.

2.2.1. Initialization of the CIOA

In the CIOA, each search agent moves along arcs governed by two main parameters: An angle θ defined by the user and a radius r calculated by the algorithm, whose value depends on the evaluation of the objective function: the better the evaluation of the objective function performed by a given search agent, the lower the value of r for this agent in the next iteration.

The CIOA is initialized generating a vector $\vec{r}$ in which the value of each element r_j is calculated by Eq. (1).

$$r_j = \frac{c_r . j^2}{N_{ag}}, \quad 1 \le j \le N_{ag} \tag{1}$$

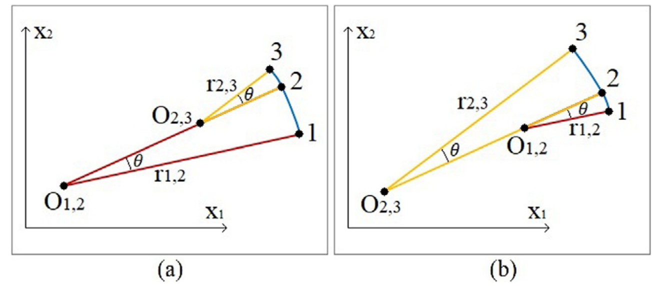


Fig. 1. Changing radius and updating the center of the circle: (a) the search agent improves its classification by reducing the radius size, (b) the search agent worsens its classification by increasing the radius size.

in which N_{ag} is the number of search agents and c_r is a constant determined using Eq. (2). In this way, the elements of the radius vector are ordered in ascending order, in which the first element corresponds to $r_1 = c_r/N_{ag}$ and the last element is given by $r_{N_{ag}} = c_r.N_{ag}$.

$$c_r = \frac{\sqrt{U_b - L_b}}{N_{ag}} \tag{2}$$

in which U_b and L_b are the upper and lower bounds of each variable, respectively.

After each search agent assigns random values to the design variables in the first solution, an evaluation of the objective function is carried out and then each search agent is classified in a ranking according to the quality of the solution that it obtained.

2.2.2. CIOA main loop

Throughout the iterations, each search agent will have its coordinates in the next iteration defined by the classification performed in the previous iteration. The best-classified agents, i.e., those who obtained the best solutions will make shorter movements (using smaller radii of the vector $\vec{r}$) while the agents that occupy the worst classifications will make longer movements. A search agent classified as the jth best solution in an iteration k will have its new coordinates in the iteration $k+1$ calculated using Eqs. (3) and (4).

$$x_{2i}(k+1) = x_{2i}(k) - rand_1.r_j.\sin(k.\theta) + rand_2.r_j.\sin((k+1).\theta) \tag{3}$$

$$x_{2i-1}(k+1) = x_{2i-1}(k) - rand_3.r_j.\cos(k.\theta) + rand_4.r_j.\cos((k+1).\theta) \tag{4}$$

in which $2i$ and $2i-1$ refer to even and odd numbers, respectively. Therefore Eq. (3) updates the $\vec{x} = [x_2; x_4; x_6; \ldots]$ coordinates while Eq. (4) updates the $\vec{x} = [x_1; x_3; x_5; \ldots]$ coordinates; *rand* variables are random numbers with a uniform distribution between zero and 1; the angle θ is a parameter provided by the user; the variable r_j corresponds to the jth element of the vector $\vec{r}$, i.e., the jth smallest radius.

The process of changing radii and updating the center of the circle is outlined below: In a hypothetical case in which all random variables have the same value, it is assumed that in an iteration a search agent departs from point 1 to point 2 in a movement governed by angle θ and radius $r_{1,2}$ with center at $O_{1,2}$. In the next iteration, the agent leaves point 2 and moves to point 3 maintaining the angle θ, the movement is governed by a radius $r_{2,3}$ and has a center at $O_{2,3}$. Two possible distinct cases of this process are illustrated in Fig. 1: The first in (a), in which the agent improves its classification by reducing the radius size, i.e., $r_{2,3} < r_{1,2}$; the second illustrated in (b), in which the search agent worsens its classification by increasing the radius

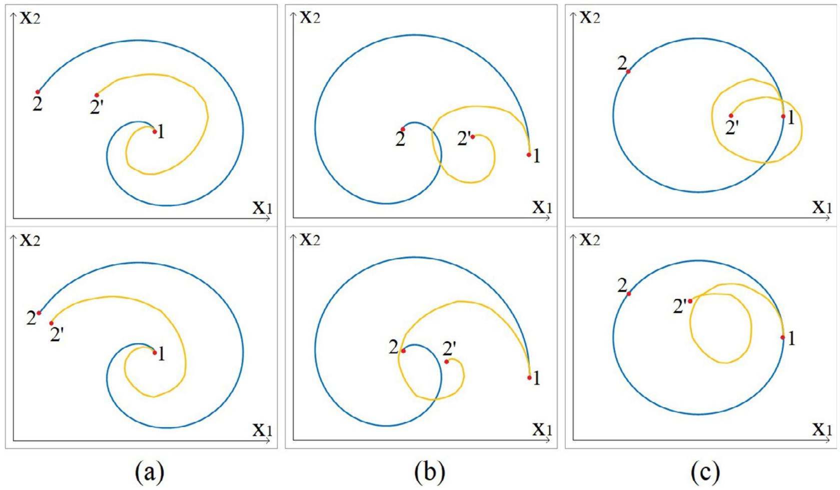


Fig. 2. Behavior of a search agent in extreme situations: (a) Case 1, (b) Case 2, (c) Case 3. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

size, i.e., $r_{2,3} > r_{1,2}$. To facilitate understanding, the action of random variables was not considered in Fig. 1.

Over consecutive iterations, each agent is likely to alternate their performance at random in the ranking of the best positions. But three hypothetical cases of behavior can occur:

(a) In the first case, a search agent always worsens its classification, increasing the radius that governs its movement along with the iterations;

(b) In the second case, a search agent always improves its classification, reducing the radius that governs its movement along with the iterations;

(c) In the third case, a search agent always presents the same classification among the group of agents, keeping their radius constant.

Fig. 2 illustrates the behavior of a search agent in each of the hypothetical extreme situations applied to a two-dimensional problem: in (a) the behavior presented in Case 1 is described, in (b) the behavior exposed in Case 2 and in (c) the behavior described in Case 3. In each frame of Fig. 2, the agent starts its journey in position 1. The blue line represents the agent's trajectory without including randomization while the yellow line represents one of the possible trajectories when randomization is considered (variables rand in Eqs. (3) and (4)). In this way, the agent ends its journey in position 2 when randomization is not considered and in position 2' when considering random variables. Two examples are presented for each extreme case, in which, in each case, it is observed that without randomization the behavior of a given agent is the same in both examples, whereas when considering random variables, the behavior tends to change each execution.

In the hypothesis that in a given iteration a variable x_i has a value greater than U_b or less than L_b this variable will receive the value corresponding to the variable x_i of the search agent who obtained the best solution in the most recent iteration.

Whenever after k iterations the search agents complete a complete lap, that is, whenever $k.\theta$ exceeds a multiple of 360° an update of the vector $\vec{r}$ is calculated using Eq. (5), to accelerate the convergence of the algorithm.

$$\vec{r}_{new} = \vec{r}.0.99 \tag{5}$$

in which $\vec{r}_{new}$ is the new radii vector after the update.

2.2.3. Local search phase

The main loop of the CIOA formulated in the previous section promotes global and local searches simultaneously. This is because the agents that produce the best solutions describe small movements corresponding to a local search while the agents with the worst solutions describe large movements corresponding to a global search. However, it is important that the algorithm dedicates some iterations only to local search, in which all agents are restricted to the most promising areas of the search space. A parameter $Glob_{It}$ that represents the proportion of iterations before the exclusively local search is inserted in the algorithm and its value can vary freely in the interval (0, 1]. However, the use of $0.75 \leq Glob_{It} \leq 0.95$ is recommended.

The exclusively local search will start at iteration k when the ratio between k and the total number of iterations is greater than $Glob_{It}$. At this moment, all search agents are restarted assuming the coordinates that produced the best solution so far. In addition, a change is made to the upper and lower bounds of each variable so that the new limits are given by U_{b1_i} and L_{b1_i}, calculated using Eqs. (6) and (7).

$$U_{b1_i} = x_{i_{best}} + \frac{U_b - L_b}{10000} \tag{6}$$

$$L_{b1_i} = x_{i_{best}} - \frac{U_b - L_b}{10000} \tag{7}$$

in which $x_{i_{best}}$ is the variable in dimension i that produced the best solution so far.

After initializing, this local search step is governed by the same equations as the main loop described in Section 2.2.2, replacing U_b with U_{b1_i} and L_b with L_{b1_i}. In this way, the search space for agents is restricted, forcing the values for the design variables to be close to the values that generated the best solution in the main loop. For the specific cases in which $L_{b1_i} < L_b$ or $U_{b1_i} > U_b$, whenever the value of a design variable is in the ranges $L_{b1_i} < x_i < L_b$ or $U_{b1_i} > x_i > U_b$, its value will be automatically updated to, respectively, $x_i = L_b$ or $x_i = U_b$.

2.2.4. User information and pseudocode

In this section, the most important information that should be taken into account by CIOA users is added, to take the best advantage of the proposed algorithm. The suggested values for

Algorithm 1 Circle-Inspired Optimization Algorithm

Begin
  Define θ and $Glob_{It}$
  Initialize a radii vector $\vec{r}$ by Eq. 1
  Assign random values to design variables
  Evaluate the objective function for each search agent
  While 1 ($k \leq Glob_{It} \times$ Maximum number of iterations)
    Classify search agents when the quality of the solution obtained
    Update de position of agents using Eqs. 3 and 4
    Verify that a search agent's design variable exceeds the limits imposed
    If k is a multiple of the value rounded down from $360/\theta$
      Update $\vec{r}$ by Eq. 5
    End If
  End While 1
  Assign to all agents the position that has generated the best solution so far
  Update the range of variables by Eqs. 6 and 7
  While 2 ($k \leq$ Maximum number of iterations)
    Repeat the procedures described in While 1, using the new range of variables
  End While 2
  Results visualization
End

the CIOA parameters were obtained through sensitivity analysis of the algorithm.

(a) Define $0.75 < Glob_{It} < 0.95$. To promote a good balance between accuracy and computational time $Glob_{It} = 0.85$ is recommended. Larger values can reduce computational time while smaller values tend to increase accuracy.

(b) Although θ can take on any value between 0° and 360°, tests carried out on several problems show that, for better performance, θ should not be a 360° divider. Relatively low θ values, such as $\theta = 13°$, $\theta = 17°$ or $\theta = 19°$, produced good results in different types of problems, with different numbers of agents and iterations considered. High values of θ produce excellent results only in specific cases, presenting inferior performance in structural optimization problems, for example.

(c) The algorithm performs better when the number of iterations is 2 to 5 times greater than the number of search agents. Preferably use more than 100 search agents.

The CIOA pseudocode is presented in Algorithm 1.

2.3. Software functionalities

In the Main_Program_CIOA.m file, the user will have to define the θ angle, in degrees, and the $Glob_{It}$ parameter (*ThetaAngle* and *GlobIt* in the code, as shown in Fig. 3(a). In addition, the following must be informed: The number of variables in the problem (N_{var}); the number of search agents and iterations to be considered (N_{ag} and N_{it}) and the upper and lower limits for the design variables (U_b and L_b). In the Objective_Function_CIOA.m file, the user will describe the objective function (*obj*) and the number of equality and inequality constraints (*nug* and *nuh*). If there are no restrictions, the number informed will be zero. Then, the restrictions must be implemented, as well as the weight value of the penalty to be considered, as shown in Fig. 3(b). For use, the code must be executed in the Main_Program_CIOA.m.

3. Illustrative examples

In this section, the Circle-Inspired Optimization Algorithm (CIOA) is validated through an experimental analysis involving the optimization of ten benchmark functions known from the literature. Then, the CIOA is applied in the solution of five real-world problems used in CEC 2020 and also in engineering problems related to structural optimization well known in the literature. Finally, statistical and convergence analyses are performed.

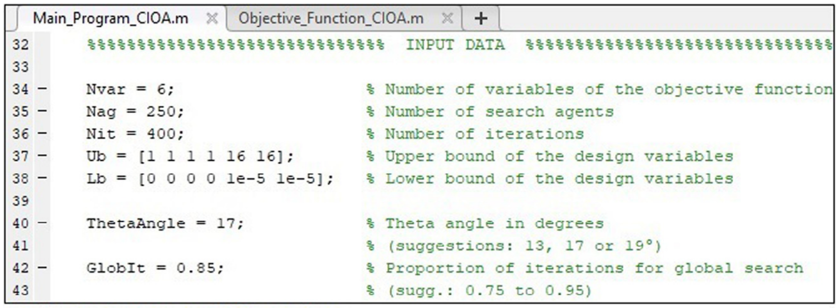
In all analyses, the performance of the CIOA is compared to that of five famous algorithms: PSO [3], HS [5], FA [9,10], SGA [18] and WOA [19]. The algorithms are implemented according to their original papers in Matlab using the following computing platform: Windows 10, 8th generation Core i5 processor and 8 GB of memory. In all simulations, the parameters used for the CIOA take into account the instructions presented in Section 2.2.4that were obtained through sensitivity analyses. Therefore, it was adopted for the CIOA $\theta = 17°$ and $Glob_{It} = 0.85$.

3.1. Experimental analysis through benchmark functions

Ten functions are used to validate the CIOA, which are presented in Table 1. These functions are part of a well-known set of benchmark test functions used to validate metaheuristic algorithms [17,19–24,27–32,34]. These functions are divided into three groups. The functions of the first group, f_1 to f_3, are unimodal and evaluate the exploitation capacity of the algorithm. The functions of the second and third groups determine the exploration capability of the algorithm. The second group contains multimodal functions (f_4 and f_5), while the third group is formed by multimodal functions with fixed dimensions (f_6 to f_{10}).

For each problem, 50 independent runs were performed. In each run, 200,000 objective function evaluations were executed, consisting of 250 research agents and 800 iterations. The results obtained, shown in Table 2, are the best value for the global optimum, the mean value for the global optimum, the standard deviation between runs and the average computational time for each run (in seconds).

As can be seen in Table 2, for the optimal value of the objective function, the CIOA presented the best result in one multimodal function (f_5) and in all five multimodal functions with fixed dimensions (f_6 to f_{10}). For the results in terms of mean value, CIOA performed best on a unimodal function (f_2), a multimodal function (f_5) and all five multimodal functions with fixed dimensions (f_6 to f_{10}). Regarding computational time, the CIOA presented the second best performance in all functions. More detailed statistical analyses are presented in Section 3.3.

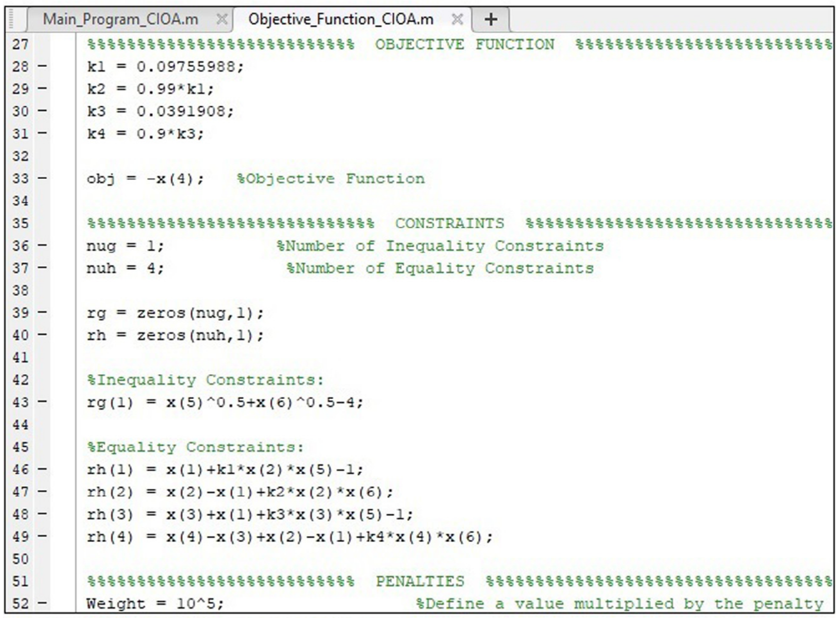


```
Main_Program_CIOA.m   Objective_Function_CIOA.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  INPUT DATA  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Nvar = 6;                     % Number of variables of the objective function
Nag = 250;                    % Number of search agents
Nit = 400;                    % Number of iterations
Ub = [1 1 1 1 16 16];         % Upper bound of the design variables
Lb = [0 0 0 0 1e-5 1e-5];     % Lower bound of the design variables

ThetaAngle = 17;              % Theta angle in degrees
                              % (suggestions: 13, 17 or 19°)
GlobIt = 0.85;                % Proportion of iterations for global search
                              % (sugg.: 0.75 to 0.95)
```

(a) Main_Program_CIOA.m file



```
Main_Program_CIOA.m   Objective_Function_CIOA.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%  OBJECTIVE FUNCTION  %%%%%%%%%%%%%%%%%%%%%%%%%%%
k1 = 0.09755988;
k2 = 0.99*k1;
k3 = 0.0391908;
k4 = 0.9*k3;

obj = -x(4);   %Objective Function

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  CONSTRAINTS  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nug = 1;           %Number of Inequality Constraints
nuh = 4;            %Number of Equality Constraints

rg = zeros(nug,1);
rh = zeros(nuh,1);

%Inequality Constraints:
rg(1) = x(5)^0.5+x(6)^0.5-4;

%Equality Constraints:
rh(1) = x(1)+k1*x(2)*x(5)-1;
rh(2) = x(2)-x(1)+k2*x(2)*x(6);
rh(3) = x(3)+x(1)+k3*x(3)*x(5)-1;
rh(4) = x(4)-x(3)+x(2)-x(1)+k4*x(4)*x(6);

%%%%%%%%%%%%%%%%%%%%%%%%%%%  PENALTIES  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Weight = 10^5;                    %Define a value multiplied by the penalty
```

(b) Objective_Function_CIOA.m file

Fig. 3. CIOA functionalities.

Table 1
Benchmark test functions: D is the number of design variables and f_{min} is the optimal value.

Group	Function	D	Range	f_{min}
Unimodal	$f_1(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	10	$[-10, 10]$	0
	$f_2(x) = \sum_{i=1}^{D-1} [100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2]$	10	$[-30, 30]$	0
	$f_3(x) = \sum_{i=1}^{D} (\lvert x_i + 0.5 \rvert)^2$	10	$[-100, 100]$	0
Multimodal	$f_4(x) = \sum_{i=1}^{D} \left[x_i^2 - 10cos\,(2\pi x_i) + 10\right]$	10	$[-5.12, 5.12]$	0
	$f_5(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	10	$[-600, 600]$	0
Multimodal with fixed dimensions	$f_6(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	$[-5, 5]$	0.00030
	$f_7(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	6	$[0, 1]$	-3.32
	$f_8(x) = -\sum_{i=1}^{5} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	-10.1532
	$f_9(x) = -\sum_{i=1}^{7} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	-10.4028
	$f_{10}(x) = -\sum_{i=1}^{10} \left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	4	$[0, 10]$	-10.536

Table 2
Results for the benchmark functions.

F		CIOA	WOA	SGA	FA	HS	PSO
f_1	best	2.74E−05	**0.00E+00**	1.10E−03	6.86E−02	1.73E−04	**0.00E+00**
	mean	1.77E−04	**0.00E+00**	1.39E−03	8.31E−02	2.26E−04	**0.00E+00**
	std	1.32E−04	**0.00E+00**	1.86E−04	1.04E−02	6.12E−05	**0.00E+00**
	time	3.19E+00	5.63E+00	**2.00E+00**	7.09E+00	1.10E+01	9.60E+00
f_2	best	5.48E−03	**1.87E−03**	1.11E+00	3.59E+00	7.87E−02	3.05E−01
	mean	**2.81E+00**	4.65E+00	3.13E+00	8.50E+00	5.91E+00	2.86E+00
	std	**1.74E+00**	9.74E+00	6.46E−01	3.11E+00	6.33E+00	1.77E+00
	time	3.32E+00	5.86E+00	**2.28E+00**	7.46E+00	1.11E+01	1.01E+01
f_3	best	6.67E−10	**0.00E+00**	1.97E−05	4.63E−02	4.93E−09	**0.00E+00**
	mean	4.01E−09	**0.00E+00**	2.90E−05	6.39E−02	7.94E−09	**0.00E+00**
	std	2.64E−09	**0.00E+00**	8.42E−06	1.05E−02	3.59E−09	**0.00E+00**
	time	3.07E+00	5.59E+00	**2.00E+00**	7.12E+00	1.09E+01	9.40E+00
f_4	best	1.61E−06	**0.00E+00**	1.15E−05	2.91E+00	9.09E−07	3.98E+00
	mean	3.45E−01	6.53E+00	1.69E+00	9.00E+00	**1.42E−06**	1.06E+01
	std	4.68E−01	7.15E+00	1.03E+00	3.64E+00	**3.23E−07**	4.33E+00
	time	3.27E+00	5.76E+00	**2.17E+00**	7.27E+00	1.11E+01	9.47E+00
f_5	best	**1.41E−07**	7.58E−02	9.13E−05	1.82E−01	4.18E−02	4.18E−02
	mean	**5.39E−07**	2.00E−01	1.59E−03	3.58E−01	7.52E−02	8.89E−02
	std	**7.89E−07**	1.12E−01	6.56E−03	7.03E−02	2.22E−02	5.02E−02
	time	3.42E+00	5.73E+00	**2.33E+00**	7.52E+00	1.09E+01	9.52E+00
f_6	best	**3.07E−04**	3.07E−04	3.32E−04	3.08E−04	**3.07E−04**	**3.07E−04**
	mean	**3.16E−04**	6.55E−04	6.00E−04	3.40E−04	4.62E−04	4.21E−04
	std	**5.95E−06**	4.49E−04	1.19E−04	7.78E−05	3.53E−04	3.12E−04
	time	3.59E+00	4.53E+00	**3.27E+00**	2.01E+01	9.74E+00	1.24E+01
f_7	best	−**3.3220**	−**3.3220**	−**3.3220**	−**3.3220**	−**3.3220**	−**3.3220**
	mean	−**3.3220**	−3.2315	−3.2055	−3.2744	−3.2792	−3.2578
	std	**2.91E−09**	5.67E−02	1.68E−02	5.88E−02	5.76E−02	5.99E−02
	time	4.75E+00	5.43E+00	**3.44E+00**	2.02E+01	1.04E+01	1.12E+01
f_8	best	−**10.1532**	−**10.1532**	−**10.1532**	−10.1529	−**10.1532**	−**10.1532**
	mean	−**10.1532**	−9.8473	−10.1532	−10.1518	−4.1063	−6.2816
	std	**5.59E−09**	1.22E+00	1.92E−07	5.81E−04	2.76E+00	3.21E+00
	time	4.67E+00	5.54E+00	**4.39E+00**	2.16E+01	1.07E+01	1.34E+01
f_9	best	−**10.4029**	−**10.4029**	−**10.4029**	−10.4028	−**10.4029**	−**10.4029**
	mean	−**10.4029**	−9.6588	−10.4029	−10.4017	−5.8754	−8.9268
	std	**5.87E−09**	1.86E+00	1.73E−07	6.61E−04	3.63E+00	2.72E+00
	time	5.07E+00	5.83E+00	**4.64E+00**	2.12E+01	1.12E+01	1.41E+01
f_{10}	best	−**10.5364**	−**10.5364**	−**10.5364**	−10.5360	−**10.5364**	−**10.5364**
	mean	−**10.5364**	−10.1038	−10.5364	−10.5352	−6.9520	−8.5055
	std	**5.46E−09**	1.48E+00	2.40E−07	5.19E−04	3.79E+00	3.33E+00
	time	5.84E+00	6.65E+00	**5.40E+00**	2.20E+01	1.17E+01	1.45E+01

Table 3
Real-World and Structural Engineering Optimization Problems: D is the number of variables, n_g the number of inequality constraints, and n_h the number of equality constraints.

Problem	Name or Description	D	n_g	n_h
R_1	Reactor Network Design (RND) [63]	6	1	4
R_2	Two-reactor Problem [63]	7	4	4
R_3	Tension/compression spring design [63]	3	3	0
R_4	Pressure vessel design [63]	4	4	0
R_5	Planetary gear train design optimization problem [63]	9	10	1
S_1	Size optimization of a 25-bar space truss with stress and displacement constraints [46,49,53,54]	8	124	0
S_2	Shape and size optimization of an 18-bar plane truss with stress and buckling constraints [47,49,53]	12	54	0
S_3	Shape and size optimization of a 52-bar space truss with multiple natural frequency constraints [48,50–53]	13	2	0
S_4	Size optimization of a realistic transmission tower of a 163-bar with stress, displacement, buckling and fundamental natural frequency constraints [53]	11	810	0

Table 4
Results obtained for the Real-World Problems and Structural Engineering Problems.

Prob.		CIOA	WOA	SGA	FA	HS	PSO
R_1	*BV*	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	best	**−0.375348**	−0.335389	−0.012813	−0.270118	−0.369001	−0.367176
	MV	2.48E−07	1.23E−04	0.00E+00	9.84E−05	**0.00E+00**	1.18E−06
	mean	−0.337139	−0.153401	−0.004673	−0.248812	**−0.334576**	−0.311166
	std	1.66E−02	1.68E−01	**3.15E−03**	9.65E−02	2.58E−02	5.65E−02
	time	1.84	2.49	**1.48**	9.73	5.28	5.90
R_2	*BV*	**0.00E+00**	0.00E+00	0.00E+00	7.66E−04	1.70E−08	1.25E−01
	best	**100.0198**	100.2392	100.1649	99.2610	103.5389	136.8535
	MV	**2.00E−02**	7.07E−02	7.50E−02	1.20E−01	1.10E−01	1.27E−01
	mean	**118.3750**	137.9467	132.6793	145.6808	151.0005	151.6865
	std	**8.89E+00**	2.62E+01	2.77E+01	1.79E+01	2.38E+01	1.49E+01
	time	2.05	2.79	**1.45**	9.59	4.74	5.05
R_3	*BV*	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	best	**0.0126654**	0.0126679	0.0127192	0.0126914	0.0126673	0.0126688
	MV	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	mean	**0.0126966**	0.0128988	0.0127197	0.0127583	0.0155228	0.0133129
	std	2.56E−05	3.02E−04	**4.17E−07**	7.24E−05	2.09E−03	5.15E−04
	time	1.46	1.63	**1.26**	9.72	3.62	4.96
R_4	*BV*	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	best	**6059.713**	6060.718	6059.975	6060.814	6059.715	6090.526
	MV	**0.00E+00**	2.35E−07	9.65E−09	0.00E+00	0.00E+00	0.00E+00
	mean	**6118.333**	6217.558	6150.204	6248.245	6990.840	6380.185
	std	**1.02E+02**	3.50E+02	1.25E+02	1.95E+02	4.15E+02	2.55E+02
	time	1.69	1.92	**1.35**	10.57	4.22	5.31
R_5	*BV*	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	best	**0.523250**	0.525730	0.526281	0.526281	0.530000	0.529296
	MV	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	mean	**0.529220**	0.534866	0.530465	0.530287	0.560294	0.535737
	std	**2.58E−03**	5.26E−03	3.14E−03	4.02E−03	2.93E−02	5.97E−03
	time	3.46	3.33	**1.70**	7.61	6.30	5.66
S_1	best	247.294	247.409	**247.275**	247.440	247.953	247.387
	mean	247.315	247.632	**247.281**	248.465	248.713	247.748
	std	2.45E−02	1.90E−01	**6.89E−03**	1.21E+00	6.64E−01	3.97E−01
	time	82.82	87.88	**79.83**	108.38	98.74	103.59
S_2	best	2075.366	2083.760	**2062.459**	2118.526	2109.862	2156.886
	mean	2146.280	2138.049	**2092.427**	2269.436	2233.645	2244.520
	std	38.273	53.986	**33.868**	100.388	106.119	64.468
	time	240.72	243.62	**218.70**	246.71	257.88	263.29
S_3	best	195.429	198.459	**192.776**	199.535	200.105	202.131
	mean	**196.857**	201.455	199.507	202.976	204.241	207.010
	std	**1.574**	2.208	3.257	3.918	3.321	5.503
	time	584.91	603.38	**563.97**	635.08	630.97	722.98
S_4	best	**17485.043**	17530.890	17495.760	17506.285	17509.516	17531.429
	mean	**17521.203**	17625.222	17580.862	17657.104	17576.885	17610.460
	std	**25.413**	68.463	86.463	212.133	57.747	56.191
	time	2379.09	2238.69	**2230.02**	2473.72	2441.72	2413.71

3.2. Application in real-world problems and also in structural engineering problems

In this section, the CIOA is applied to solve real-life optimization problems. The problems addressed are presented in Table 3 and are divided into two groups: Real-world optimization problems provided for the 'CEC2020 One Goal Restricted Optimization Competition in the Real World' (Problems R_1 to R_5) and Structural optimization problems for trusses subject to multiple constraints (Problems S_1 to S_4).

Details on the implementation of real-world problems (R_1 to R_5) can be seen in [63]. Some of these problems also have equality constraints in addition to inequality constraints. In real-world problems, due to their high complexity, it is common for constraints to be violated during the optimization process. In these cases, the algorithm comparison is as follows:

(a) For each run, a constraint violation rate and the corresponding objective function value linked to this rate are calculated. Eq. (8) presents the calculation of the violation rate according to the inequality g_i and equality h_j constraints.

$$Viol = \frac{\sum_{i=1}^{n_g} \max\left(g_i\left(x\right), 0\right) + \sum_{j=1}^{n_h} \max\left(\left|h_j\left(x\right)\right| - 0.0001, 0\right)}{n_g + n_h} \tag{8}$$

(b) The best result of the objective function is obtained in the run that generated the lowest violation rate (BV). If all runs generate equal violation rates, the best result is obtained by the lowest objective function value.

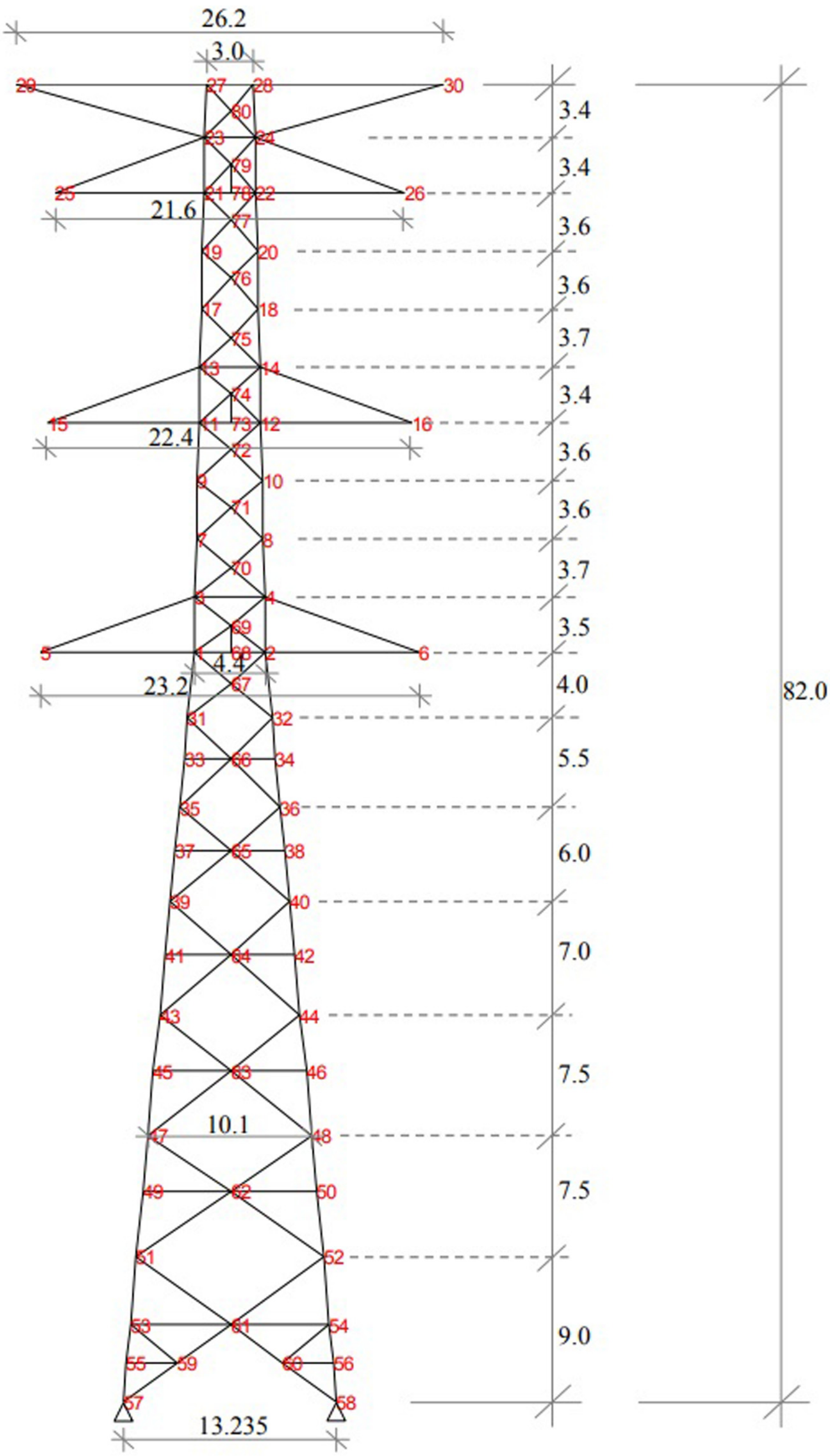


Fig. 4. Problem S_4: Realistic Tower (dimensions in meters).

(c) The mean result for the objective function value is linked to the mean of the violations of each execution (MV), according to Eq. (9), in which $Viol_k$ is the violation of each execution and n_{runs} is the number of executions.

$$MV = \frac{\sum_{k=1}^{n_{runs}} Viol_k}{n_{runs}} \tag{9}$$

(d) One algorithm is better than another whenever its constraint violation rates are lower. If two algorithms have the same rates, the best algorithm is the one with the lowest value for the objective function.

Three of the four truss optimization problems (S_1 to S_3) are well-known in the literature and have already been studied by several authors [46–54]. The S_4 optimization problem addresses the design of a real structure of an 82m-high transmission line tower which was damaged during a typhoon in Japan in 1991 [64]. Thus, the proposed optimization problem consists of minimizing the mass of the structure considering the wind effects

Table 5
Results of the Wilcoxon Signed-Rank Test for the benchmark functions.

F		CIOA × WOA	CIOA × SGA	CIOA × FA	CIOA × HS	CIOA × PSO
f_1	T^+	0	1275	1275	918	0
	T^-	1275	0	0	357	1275
	winner	WOA	CIOA	CIOA	CIOA	PSO
f_2	T^+	763	756	1273	903	650
	T^-	512	519	1	372	625
	winner	CIOA	CIOA	CIOA	CIOA	CIOA
f_3	T^+	0	1275	1275	1225.5	0
	T^-	1275	0	0	49.5	1275
	winner	WOA	CIOA	CIOA	CIOA	PSO
f_4	T^+	1044	1182	1275	0	1275
	T^-	231	93	0	1275	0
	winner	CIOA	CIOA	CIOA	HS	CIOA
f_5	T^+	1275	1275	1275	1275	1275
	T^-	0	0	0	0	0
	winner	CIOA	CIOA	CIOA	CIOA	CIOA
f_6	T^+	779	1275	490	405	329
	T^-	496	0	785	820	946
	winner	CIOA	CIOA	FA	CIOA	PSO
f_7	T^+	1184	1274	1275	1275	999
	T^-	91	1	0	0	276
	winner	CIOA	CIOA	CIOA	CIOA	CIOA
f_8	T^+	147	1275	1275	1255	1085
	T^-	1128	0	0	20	190
	winner	WOA	CIOA	CIOA	CIOA	CIOA
f_9	T^+	329	1275	1275	1024.5	534
	T^-	946	0	0	151.5	741
	winner	WOA	CIOA	CIOA	CIOA	PSO
f_{10}	T^+	194	1275	1275	908	609
	T^-	1081	0	0	317	666
	winner	WOA	CIOA	CIOA	CIOA	PSO

as constraints. Details about the implementation can be seen in [53]. In Fig. 4, the realistic structure of the tower is shown.

For problems R_1 to R_5, 25 independent runs were performed: In each round, 100,000 objective function evaluations were considered, formed through 400 iterations and 250 search agents. For problems S_1 to S_4, 10 independent runs were considered: In problems S_1 and S_4, 200,000 objective function evaluations were considered in each algorithm. In problems S_2 and S_3, with shape optimization, 400,000 and 600,000 objective function evaluations were considered, respectively. Results are presented in Table 4.

Table 4 shows the excellent performance of the CIOA for real-life optimization problems. In Engineering problems R_1 to R_5, CIOA always presented the best optimal result and did not violate the constraints in the best run (i.e., $BV = 0$). In addition, CIOA has the best mean in four of the five problems and the best standard deviation in three. In the structural optimization problems of trusses, the CIOA presented the best mean and the best standard deviation in 2 problems (S_3 and S_4), in addition, CIOA presented the best optimal design in the problem S_4, proving to be very competitive for solving problems with a high number of constraints.

3.3. Statistical and convergence analysis

In this section, statistical and convergence analyses of the CIOA algorithm are presented for the different types of optimization problems analyzed in this paper.

In many cases of comparison of algorithms, simple statistical tests such as mean and standard deviation of the results obtained by several simulations do not fully reveal which of the algorithms is the most effective. An advanced form of analysis can be done by pairwise comparison using the Wilcoxon Signed-Rank Test [16]. The detailed procedure for running the Wilcoxon Signed-Rank Test can be found in [65]. Thus, the Wilcoxon Signed-Rank Test was used to compare the CIOA with each of the other algorithms in benchmark functions analyzed in Section 3.1. The results are presented in Table 5, in which the performance of the CIOA is superior to that of the algorithm with which it was compared whenever $T^+ > T^-$.

As can be seen in Table 5, the CIOA performed better than the other algorithms in 38 of the 50 comparisons, proving the CIOA's competitiveness against rival algorithms.

Fig. 5 shows boxplots referring to the results of some problems solved in this paper. Different types of problems were selected: Functions f_1 and f_2 (unimodal), functions f_4 and f_5 (multimodal), functions f_7 and f_9 (multimodal with fixed dimensions), problems R_3 and R_5 (real-world problems for CEC 2020) and problem S_3 (Shape and size optimization of a 52-bar space truss). The presented results demonstrate the robustness of the CIOA in the analyzed optimization problems, when compared with other algorithms.

Fig. 6 shows the CIOA convergence curves for different types of optimization problems: unimodal functions (f_1 and f_3), multimodal functions (f_4 and f_5), multimodal functions with fixed dimensions (f_6 and f_8), real-world problems of CEC 2020 (R_1 and R_4) and structural optimization problems of trusses (S_1 and S_4). It is noticed that for the different types of problems the

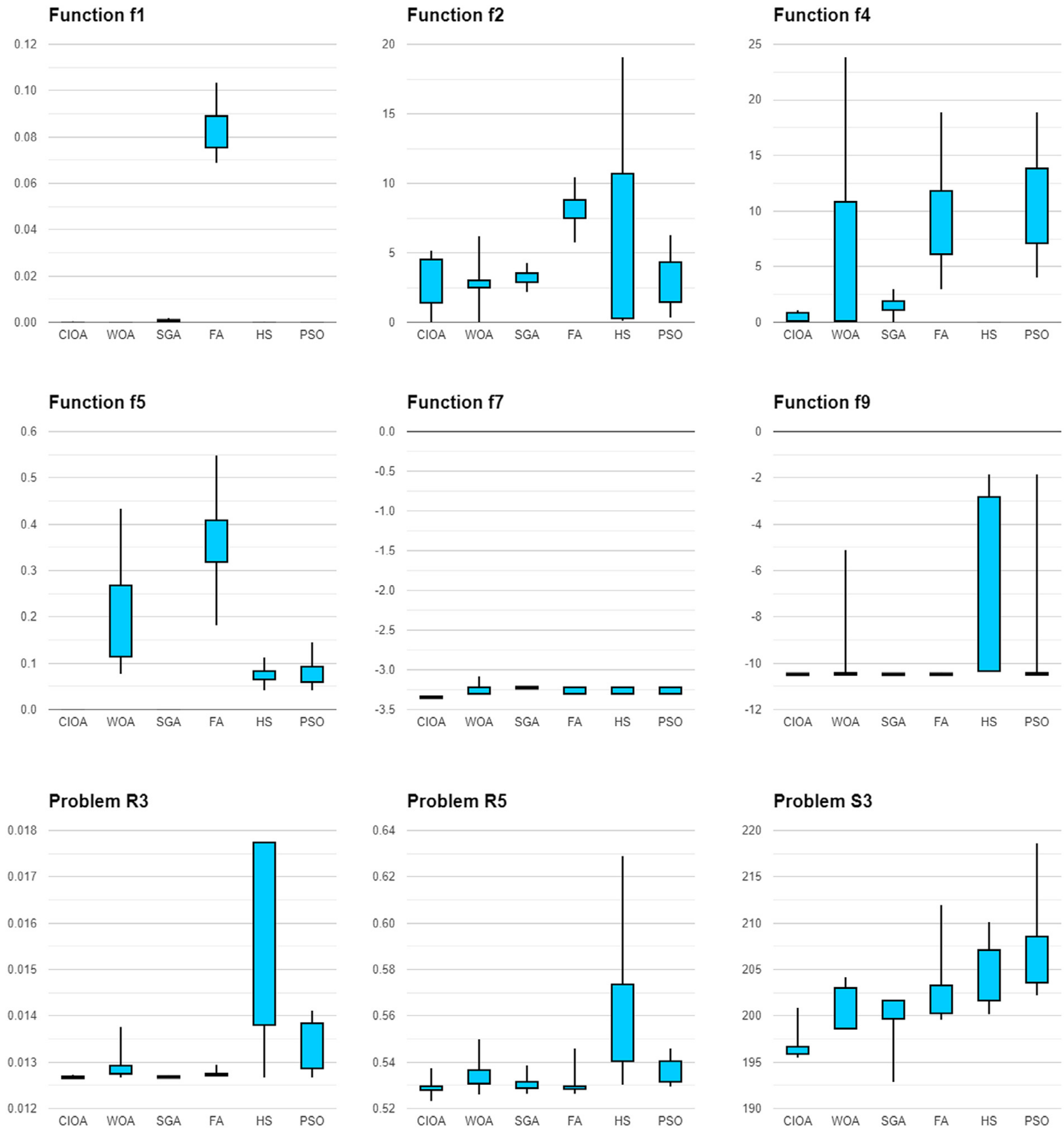


Fig. 5. Boxplots for some of the different optimization problems.

CIOA presents different convergence behaviors. In general, it is observed that the CIOA presents rapid convergence in all analyzed problems. The behavior in Problem R_1 is highlighted, which presents equality constraints that are violated at some point in the optimization process, so in later iterations there may be an increase in the objective function as long as there is a reduction in the violation rates, calculated by the CIOA.

4. Impact and conclusions

The new metaheuristic optimization algorithm presented in this paper, called Circle-Inspired Optimization Algorithm (CIOA), proved to be a powerful tool for solving complex optimization problems such as benchmark function optimization, real-world optimization problems and structural truss optimization. In direct comparisons with other algorithms, the CIOA presented the best result for global optimization in 7 of the 10 benchmark functions analyzed and in 6 of the 9 proposed application problems. Statistical analysis through Wilcoxon Signed-Rank Test and Boxplots demonstrated the superiority and robustness of the CIOA in most of the optimization problems in which it was tested. In addition, the CIOA has the advantages of fast convergence and the reduced number of parameters that must be defined by the user: only the θ angle and the $Glob_{It}$ parameter. The limitations or disadvantages of the CIOA are the high number of objective function evaluations and the number of search agents that must be used, a recurring fact in several metaheuristic algorithms. However, the CIOA's low computational operating time allows it to perform numerous operations quickly. Although the examples discussed in this paper are mainly directed to engineering optimization problems, using the CIOA to solve optimization problems in other areas of study can be promising.

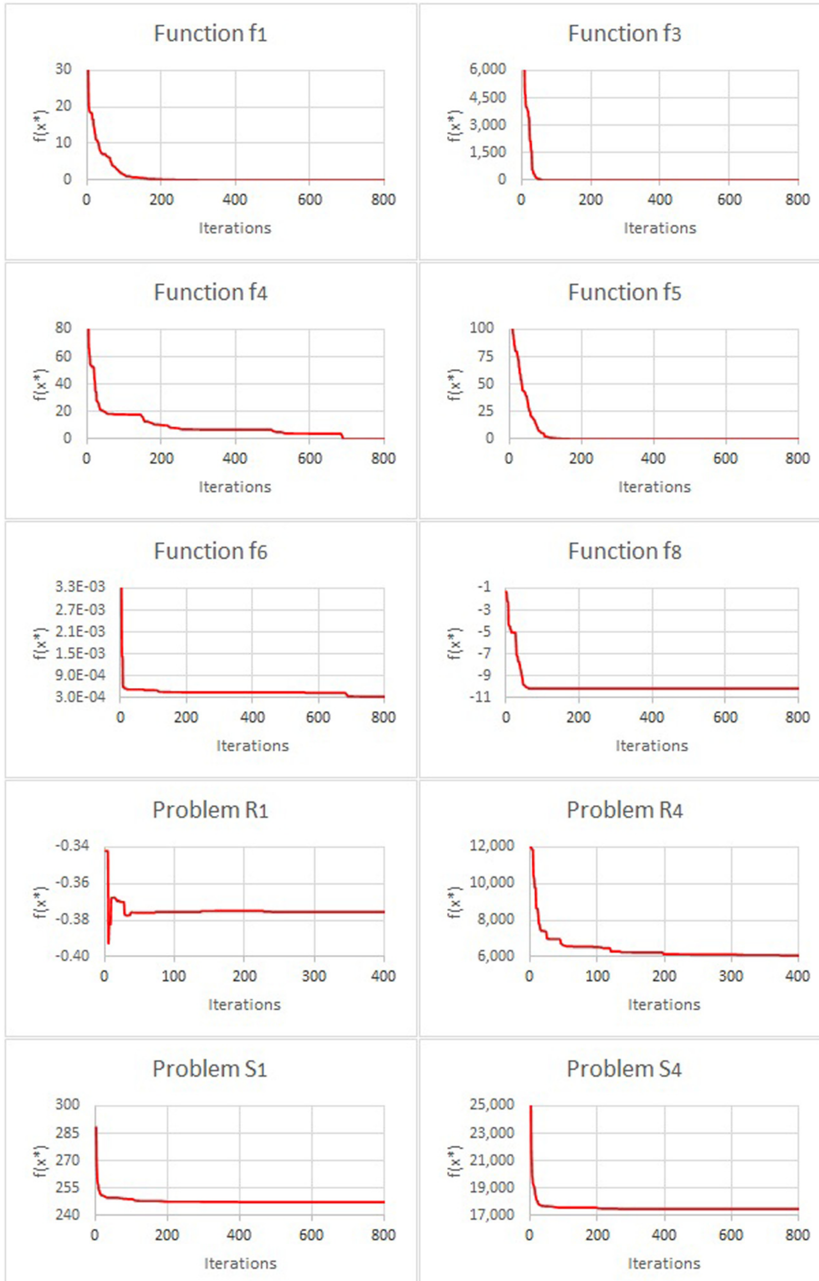


Fig. 6. CIOA convergence curves.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors acknowledge the financial support of CAPES, Brazil and CNPq, Brazil.

Appendix A. Supplementary data

The supplementary related to this article consist of the files ‘Main_Program_CIOA.m’ and ‘Objective_Function_CIOA.m’ which contain the MATLAB code of the CIOA algorithm.

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2022.101192.

References

[1] Yang XS. Nature-inspired metaheuristic algorithms. 2nd ed.. Frome: Luniver Press; 2010.

[2] S Kirkpatrick, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220:671–80. http://dx.doi.org/10.1126/science.220.4598.671.

[3] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of the international conference on neural networks. 1995, p. 1942–8. http://dx.doi.org/10.1109/ICNN.1995.488968.

[4] Storn R, Price K. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11:341–9. http://dx.doi.org/10.1023/A:1008202821328.
[5] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: Harmony search. Simulation 2001;76:60–8. http://dx.doi.org/10.1177/003754970107600201.
[6] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 2007;39:459–71. http://dx.doi.org/10.1007/s10898-007-9149-x.
[7] Karaboga D, Basturk B. Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In: Proceedings of the International Fuzzy Systems Association World Congress. 2007, p. 789–98. http://dx.doi.org/10.1007/978-3-54072950-1_77.
[8] Socha K, Dorigo M. Ant colony optimization for continuous domains. European J Oper Res 2008;185:1155–73. http://dx.doi.org/10.1016/j.ejor.2006.06.046.
[9] Yang XS. Firefly algorithms for multimodal optimization. In: Proceedings of the International Symposium on Stochastic Algorithms. 2009, p. 169–78. http://dx.doi.org/10.1007/978-3-642-04944-6_14.
[10] Yang XS. Firefly algorithm, stochastic test functions and design optimization. Int J Bio-Inspired Comput 2010;2:78–84. http://dx.doi.org/10.1504/IJBIC.2010.032124.
[11] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: A gravitational search algorithm. Inform Sci 2009;179:2232–48. http://dx.doi.org/10.1016/j.ins.2009.03.004.
[12] Yang XS, Deb S. Cuckoo search via Lévy flights. In: Proceedings of the World Congress on Nature and Biologically Inspired Computing. 2009, p. 210–4. http://dx.doi.org/10.1109/NABIC.2009.5393690.
[13] Yang XS. A new metaheuristic bat-inspired algorithm. In: Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization. 2010, p. 65–74. http://dx.doi.org/10.1007/978-3-642-12538-6_6.
[14] Yang XS. Flower pollination algorithm for global optimization. In: Proceedings of the Unconventional Computation and Natural Computation. 2012, p. 240–9.
[15] Gandomi AH, Alavi AH. Krill herd: A new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 2012;17:4831–45. http://dx.doi.org/10.1016/j.cnsns.2012.05.010.
[16] Civicioglu P. Backtracking search optimization algorithm for numerical optimization problems. Appl Math Comput 2013;219:8121–44. http://dx.doi.org/10.1016/j.amc.2013.02.017.
[17] Mirjalili S, Mirjalili SB, Lewis A. Grey wolf optimizer. Adv Eng Softw 2014;69:46–61. http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.
[18] Gonçalves MS, Lopez RH, Fadel Miguel LF. Search group algorithm: A new metaheuristic method for the optimization of truss structures. Comput Struct 2015;153:165–84. http://dx.doi.org/10.1016/j.compstruc.2015.03.003.
[19] Mirjalili S, Lewis A. The whale optimization algorithm. Adv Eng Softw 2016;95:51–67. http://dx.doi.org/10.1016/j.advengsoft.2016.01.008.
[20] Dhiman G, Kumar V. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. Adv Eng Softw 2017;114:47–70. http://dx.doi.org/10.1016/j.advengsoft.2017.05.014.
[21] Dhiman G, Kumar V. Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. Knowl-Based Syst 2018;159:20–50. http://dx.doi.org/10.1016/j.knosys.2018.06.001.
[22] Arora S, Singh S. Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput 2019;23:715–34. http://dx.doi.org/10.1007/s00500-018-3102-4.
[23] Dhiman G, Kaur A. STOA: A bio-inspired based optimization algorithm for industrial engineering problems. Eng Appl Artif Intell 2019;82:148–74. http://dx.doi.org/10.1016/j.engappai.2019.03.021.
[24] Dhiman G, Kumar V. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. Knowl-Based Syst 2019;165:169–96. http://dx.doi.org/10.1016/j.knosys.2018.11.024.
[25] Sallam KM, Elsayed SM, Chakrabortty RK, Ryan MJ. Multi-operator differential evolution algorithm for solving real-world constrained optimization problems. In: Proceedings of the IEEE congress on evolutionary computation. 2020, p. 1–8. http://dx.doi.org/10.1109/CEC48606.2020.9185722.
[26] Kumar A, Das S, Zelinda I. A self-adaptive spherical algorithm for real-world constrained optimization problems. In: Proceedings of the genetic and evolutionary computation conference companion. 2020, p. 13–4. http://dx.doi.org/10.1145/3377929.3398186.
[27] Kaur S, Awashti LK, Sangal AL, Dhiman G. Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. Eng Appl Artif Intell 2020;90:103541. http://dx.doi.org/10.1016/j.engappai.2020.103541.
[28] Dehghani M, Montazeri Z, Dehghani A, Ramirez-Mendoza RA, Samet H, Guerrero JM, et al. MLO: Multi leader optimizer. Int J Intell Eng Syst 2020;13:364–73. http://dx.doi.org/10.22266/ijies2020.1231.32.
[29] Dehghani M, Montarezi Z, Givi H, Guerrero JM, Dhiman G. Darts game optimizer: A new optimization technique based on darts game. Int J Intell Eng Syst 2020;13:286–94. http://dx.doi.org/10.22266/ijies2020.1031.26.
[30] Dehghani M, Montazeri Z, Dhiman G, Malik OP, Morales-Menendez R, Ramirez-Mendoza RA, et al. A spring search algorithm applied to engineering optimization problems. Appl Sci 2020;10:6173. http://dx.doi.org/10.3390/app10186173.
[31] Dhiman G, Garg M, Nagar A, Kumar V, Dehghani M. A novel algorithm for global optimization: Rat swarm optimizer. J Ambient Intell Humaniz Comput 2021;12:8457–82. http://dx.doi.org/10.1007/s12652-020-02580-0.
[32] Dehghani M, Montazeri Z, Malik OP, Dhiman G, Kumar V. BOSA: Binary orientation search algorithm. Int J Innov Technol Explor Eng 2019;9:5306–10. http://dx.doi.org/10.35940/ijitee.A4215.119119.
[33] Dhiman G, Oliva D, Kaur A, Singh KK, Vimal S, Sharma A, et al. BEPO: A novel binary emperor penguin optimizer for automatic feature selection. Knowl-Based Syst 2021;211:106560. http://dx.doi.org/10.1016/j.knosys.2020.106560.
[34] Dhiman G. ESA: a hybrid bio-inspired metaheuristic optimization approach for engineering problems. Eng Comput 2021;37:323–53. http://dx.doi.org/10.1007/s00366-019-00826-w.
[35] Yildiz BS, Pholdee N, Bureerat S, Yildiz AR, Sait SM. Robust design of a robot gripper mechanism using new hybrid grasshopper optimization algorithm. Expert Syst 2021;38. http://dx.doi.org/10.1111/exsy.12666.
[36] Yildiz AR, Erdas MU. A new hybrid Taguchi-salp swarm optimization algorithm for the robust design of real-world engineering problems. Mater. Testings 2021;63:157–62. http://dx.doi.org/10.1515/mt-2020-0022.
[37] Yildiz BS, Kumar S, Pholdee N, Bureerat S, Sait SM, Yildiz AR. A new chaotic Lévy flight distribution optimization algorithm for solving constrained engineering problems. Expert Syst 2022;e12992. http://dx.doi.org/10.1111/exsy.12992.
[38] Yildiz BS, Pholdee N, Bureerat S, Yildiz AR, Sait SM. Enhanced grasshopper optimization algorithm using elite opposition-based learning for solving real-world engineering problems. Eng Comput 2021. http://dx.doi.org/10.1007/s00366-021-01368-w.
[39] Gupta VK, Shukla SK, Anupriya, Rawat RS. Crime tracking system and People's safety in India using machine learning approaches. Int J Modern Res 2022;2:1–7.
[40] Vaishnav PK, Sharma S, Sharma P. Analytical review analysis for screening COVID-19 disease. Int J Modern Res 2021;1:22–9.
[41] Sharma T, Nair R, Gomathi S. Breast cancer image classification using transfer learning and convolutional neural network. Int J Modern Res 2022;2:8–16.
[42] Chatterjee I. Patenting machine-learning: Review and discussions. Int J Modern Res 2021;1:15–21.
[43] Kumar R, Dhiman G. A comparative study of fuzzy optimization through fuzzy number. Int J Modern Res 2021;2:1–14.
[44] Shukla SK, Gupta VK, Joshi K, Gupta A, Singh MK. Self-aware execution environment model (SAE2) for the performance improvement of multicore systems. Int J Modern Res 2022;2:17–27.
[45] Ghasemi MR, Hinton E, Wood RD. Optimization of trusses using genetic algorithms for discrete and continuous variables. Eng Comput 1999;16:272–301. http://dx.doi.org/10.1108/02644409910266403.
[46] Lee KS, Geem ZW. A new structural optimization method based on the Harmony Search Algorithm. Comput Struct 2004;82:781–98. http://dx.doi.org/10.1016/j.compstruc.2004.01.002.
[47] Lee KS, Geem ZW. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods Appl Mech Engrg 2005;194:3902–33. http://dx.doi.org/10.1016/j.cma.2004.09.007.
[48] Lingyun W, Mei Z, Guangming W, Guang M. Truss optimization on shape and sizing with frequency constraints based on genetic algorithm. Comput Mech 2005;35:361–8. http://dx.doi.org/10.1007/s00466-004-0623-8.
[49] Lamberti L. An efficient simulated annealing algorithm for design optimization of truss structures. Comput Struct 2008;86:1936–53. http://dx.doi.org/10.1016/j.compstruc.2008.02.004.
[50] Gomes HM. Truss optimization with dynamic constraints using a particle swarm algorithm. Expert Syst Appl 2011;38:957–68. http://dx.doi.org/10.1016/j.eswa.2010.07.086.
[51] Gomes HM. A firefly metaheuristic structural size and shape optimisation with natural frequency constraints. Int J Metaheuristics 2012;2:38–55. http://dx.doi.org/10.1504/IJMHEUR.2012.048215.
[52] Miguel LFF, Fadel Miguel LF. Shape and size optimization of truss structures considering dynamic constraints through modern metaheuristic algorithms. Expert Syst Appl 2012;39:9458–67. http://dx.doi.org/10.1016/j.eswa.2012.02.113.
[53] Miguel LFF, Fadel Miguel LF. Assessment of modern metaheuristic algorithms – HS ABC and FA – in shape and size optimization of structures with different types of constraints. Int J Metaheuristics 2013;2:256–93. http://dx.doi.org/10.1504/IJMHEUR.2013.056404.

[54] Bekdas G, Nigdeli SM, Yang XS. Sizing optimization of truss structures using flower pollination algorithm. Appl Soft Comput 2015;37:322–31. http://dx.doi.org/10.1016/j.asoc.2015.08.037.

[55] Kaveh A, Zolghadr A. Meta-heuristic methods for optimization of truss structures with vibration frequency constraints. Acta Mech 2018;229:3971–92. http://dx.doi.org/10.1007/s00707-018-2234-z.

[56] Souza OAP, Miguel LFF. Comparison of the performance of different metaheuristic optimization algorithms. In: Proceedings of the XLI Ibero-Latin American congress on computational methods in engineering. 2020, ISSN: 2675-6269.

[57] Yildiz ABS, Pholdee N, Bureerat S, Yildiz AR, Sait SM. Sine-cosine optimization algorithm for the conceptual design of automobile components. Mater Testing 2020;62:744–8. http://dx.doi.org/10.3139/120.111541.

[58] Panagant N, Pholdee N, Bureerat S, Kaen K, Yildiz AR, Sait SM. Seagull optimization algorithm for solving real-world design optimization problems. Mater Testing 2020;62:640–4. http://dx.doi.org/10.3139/120.111529.

[59] Yildiz BS, Pholdee N, Bureerat S, Erdas MU, Yildiz AR, Sait SM. Comparision of the political optimization algorithm, the archimedes optimization algorithm and the levy flight algorithm for design optimization in industry. Mater Testing 2021;63:356–60. http://dx.doi.org/10.1515/mt-2020-0053.

[60] Yildiz BS, Patel V, Pholdee N, Sait MS, Bureerat S, Yildiz AR. Conceptual comparison of the ecogeography-based algorithm equilibrium algorithm, marine predators algorithm and slime mold algorithm for optimal product design. Mater Testing 2021;63:336–40. http://dx.doi.org/10.1515/mt-2020-0049.

[61] Gupta S, Abderazek H, Yildiz BS, Yildiz AR, Mirjalili S, Sait SM. Comparison of metaheuristic optimization algorithms for solving constrained mechanical design optimization problems. Expert Syst Appl 2021;183:115351. http://dx.doi.org/10.1016/j.eswa.2021.115351.

[62] Ho YC, Pepyne DL. Simple explanation of the no-free-lunch theorem and its implications. J Optim Theory Appl 2002;115:549–70. http://dx.doi.org/10.1023/A:1021251113462.

[63] Kumar A, Wo G, Ali MZ, Mallipeddi R. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. Swarm Evol Comput 2020;56:1–27. http://dx.doi.org/10.1016/j.swevo.2020.100693.

[64] Murotsu Y, Okada H, Shao S. Reliability-based design of transmission line structures under extreme wind loads. Struct Safety Reliab 1994;3:1675–81.

[65] Derrac J, García S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 2011;1:3–18. http://dx.doi.org/10.1016/j.swevo.2011.02.002.