Mathematical Biosciences and Engineering

MBE, 18(5): 5836–5864.
DOI: 10.3934/mbe.2021294
Received: 25 April 2021
Accepted: 22 June 2021
Published: 29 June 2021

http://www.aimspress.com/journal/MBE

Research article

CWCA: Complex-valued encoding water cycle algorithm

Guo Zhou[1], Yongquan Zhou[2,3,*], Zhonghua Tang[2] and Qifang Luo[2,3]

[1] Department of Science and Technology Teaching, China University of Political Science and Law, Beijing 100088, China
[2] College of Artificial Intelligence, Guangxi University for Nationalities, Nanning 530006, China
[3] Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning 530006, China

* **Correspondence:** Email: yongquanzhou@126.com; Tel: +8613607882594; Fax: +867713265523.

Abstract: Since the meta-heuristic water cycle algorithm (WCA) was presented, it has been used extensively in scientific computation and engineering optimization. The aims of this study are to improve the exploration and exploitation capabilities of the WCA algorithm, accelerate its convergence speed, and enhance its calculation accuracy. In this paper, a novel complex-valued encoding WCA (CWCA) is proposed. The positions of rivers and streams are divided into two parts, that is, the real part and imaginary part, and modified formulas for the new positions of rivers and streams are proposed. To evaluate the performance of the CWCA, 12 benchmark functions and four engineering examples were considered. The experimental results indicated that the CWCA had higher precision and convergence speed than the real-valued WCA and other well-known meta-heuristic algorithms.

Keywords: water cycle algorithm; complex-valued water cycle algorithm; benchmark functions; engineering optimization; meta-heuristic optimization.

1. Introduction

Generally, the objective of optimization is to find the optimal feasible response, considering the constraints of a problem. As a modern optimization method, meta-heuristic algorithms have been proposed by researchers in recent years, for example, the genetic algorithm (GA) [1], particle swarm optimization (PSO) [2], simulated annealing (SA) [3], glowworm swarm optimization (GSO) [4],

harmony search (HS) [5], bacterial foraging optimization [6], earthworm optimization [7], bat algorithm (BA) [8], elephant herding behavior algorithm [9], krill herd algorithm [10], water cycle algorithm (WCA) [10], and various improved versions of the meta-heuristic optimization algorithm [45–48]. To date, hundreds of meta-heuristics have been proposed and used successfully to solve complex optimization problems.

In 2012, Eskandar et al. presented the WCA and used it to solve real-valued optimization issues. The real-valued WCA is a meta-heuristic optimization algorithm inspired by the water cycle process in nature. It considers how rivers and streams flow into the sea. A simplified water cycle system diagram is illustrated in Figure 1. In the WCA, the initial population comprises raindrops; the best raindrop represents the ocean. Many good raindrops represent a river, and the remaining raindrops represent streams that flow to the ocean and rivers. The rivers flow to the ocean, which is the lowest terrain location.

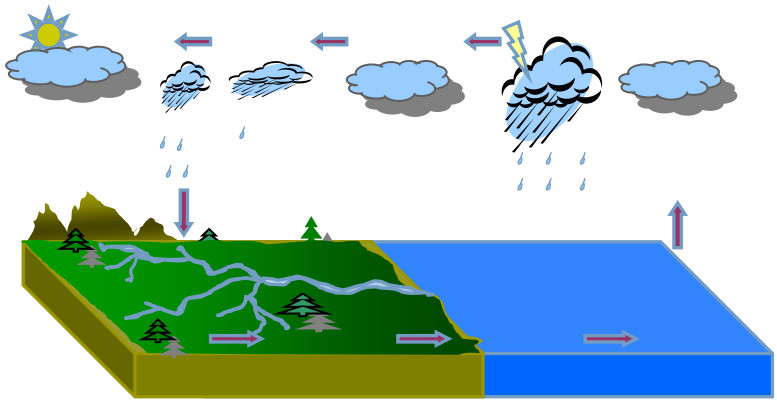

Figure 1. Simplified water cycle process.

Since the WCA was proposed, it has been used extensively in scientific computation and engineering optimization. In 2014, Ail et al. applied the WCA to multi-objective optimization [12]. Zhang et al. applied the WCA to solve engineering optimization problems [13]. In 2015, Ail et al. proposed the WCA with the evaporation rate (ERWCA) for unconstrained and constrained optimization [14]. In [41], a comprehensive and exhaustive review was conducted on the WCA and its applications in a wide variety of study fields, including mechanical engineering, electrical and electronic engineering, civil engineering, industrial engineering, water resources and hydropower engineering, computer engineering, and mathematics. In [42], the detailed open source code for the WCA was provided, and its performance and efficiency for solving optimization problems was demonstrated. In [43], an enhanced discrete version of the WCA called DWCA was proposed to solve the symmetric and asymmetric traveling salesman problem. The designed solver was tested on over 33 problem datasets, and the statistical significance of the performance gaps for this benchmark was validated using results from non-parametric tests, not only in terms of optimality but also in terms of convergence speed. In [44], an extended version of WCA, that is, gradient-based WCA (GWCA) with the evaporation rate, was introduced to enhance the performance of the standard WCA by incorporating a local optimization operator in a so-called gradient-based approach. The experimental results demonstrated the feasibility and efficiency of the proposed GWCA.

In this paper, the complex-valued encoding WCA (CWCA) is proposed, which uses the complex number coding method of the complex-valued BA [15] and individual genes in the

evolutionary algorithm [16,17]. A diploid is adopted in the indication of individual genes, which significantly enhances the individual's in information capacity. Finally, twelve benchmark functions and four engineering examples were considered to evaluate the performance of the CWCA. The test results indicated that the CWCA had higher precision and convergence speed than the real-valued WCA and other well-known meta-heuristic algorithms.

2. Mathematical model for the real-valued water cycle

In the WCA, the initial population is represented by $N_{pop} \times N$ matrix:

$$Population = \begin{bmatrix} Sea \\ River_1 \\ River_2 \\ River_3 \\ \vdots \\ Stream_{Nsr+1} \\ Stream_{Nsr+2} \\ Stream_{Nsr+3} \\ \vdots \\ Stream_{N_{pop}} \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N_{pop},1} & x_{N_{pop},2} & x_{N_{pop},3} & \cdots & x_{N_{pop},N} \end{bmatrix}, \tag{1}$$

where N_{pop} represents the population size and N represents the design variables. Initially, the N_{pop} stream is randomly generated, and N_{sr} good individuals represent a sea or river. The stream that has the minimum value among all streams is identified as the ocean. N_{sr} represents the summation of the number of rivers.

The remainder of the populations are computation using

$$N_{sr} = Number\ of\ Rivers + \underbrace{1}_{Sea} \tag{2}$$

$$N_{stream} = N_{pop} - N_{sr}\,. \tag{3}$$

The total volume of water flowing in a river and/or the ocean varies from stream to stream. The number of specified streams for a river or the ocean can be calculated as

$$NS_n = round\left\{\left|\frac{f(River_k)}{\sum_{k=1}^{N_{sr}} f(River_k)}\right| \times N_{Stream}\right\}, \quad k = 1,2,\cdots N_{sr}, \tag{4}$$

where NS_n denotes the number of rivers and f represents the fitness function.

In nature, streams are formed by raindrops, and then they join each other to constitute new rivers. A part of a streams flows directly into the ocean. All streams and rivers end up in the open ocean (the best point). Figure 2 shows a schematic of a stream flowing into a river. The star represents the river and the circles represent the stream.

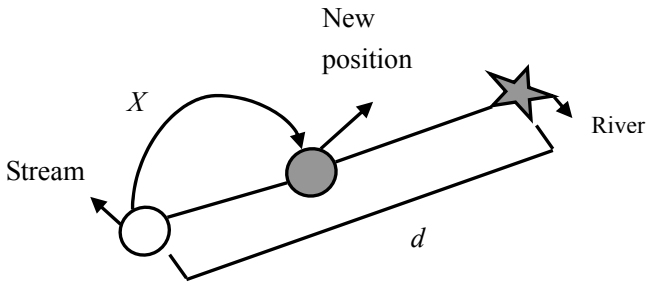

Figure 2. Schematic of a stream flowing into a river.

In Figure 2, a stream flows into a river using a random distance:

$$X \in (0, C \times d), \quad C > 1, \tag{5}$$

where C is a numerical value between 1 and 2 (near to 2). The best numerical value is chosen as 2. d represents the current distance between the stream and river current distance. X is a random number between 0 and ($C \times d$).

The position update for streams and rivers is determined

$$X_{Stream}^{k+1} = X_{Stream}^{k} + rand \times C \times (X_{River}^{k} - X_{Stream}^{k}) \tag{6}$$

$$X_{River}^{k+1} = X_{River}^{k} + rand \times C \times (X_{Sea} - X_{River}^{k}), \tag{7}$$

where *rand* represents a random number uniformly distributed between 0 and 1. If the current solution provided by a stream is better than that of its connecting river, the positions of the river and stream are swapped. Such a swap can also occur for rivers and the ocean.

New raindrops form streams in different places to avoid being trapped in local optima. The rainfall condition is as follows:

$$| X_{Ocean} - X_{\text{Revier}}^{k} | < eps, k = 1,2,..., N_{sr}, \tag{8}$$

where eps is a small numerical constant.

The stream and river location update formulas are

$$X_{Stream}^{new} = LB + rand \times (UB - LB) \tag{9}$$

$$X_{River}^{new} = LB + rand \times (UB - LB), \tag{10}$$

where LB represents the lower bound and UB represents the upper bound. The best newly formed raindrop is treated as a river running into the ocean. The remaining raindrops form new streams that flow into the rivers or ocean.

In Figure 3, the circles, stars, and diamonds represent streams, rivers, and the ocean, respectively. In view to the above description, the WCA steps are as follows:

Step 1. Initialize parameters N_{pop}, N_{sr}, and the number of maximum iterations *iterMax*.

Step 2. Randomly generate the initial population, composite initial rivers, streams, and ocean using Eqs (1)–(3).

Step 3. Calculate the intensity of streams flowing into the rivers and ocean using Eq (4).

Step 4. Determine the streams' flow into the rivers, and the rivers' flow into the ocean using Eqs (6) and (7).

Step 5. According to the fitness function value, swap the positions of the streams, rivers, and ocean.

Step 6. If the termination condition is not satisfied [using Eq (8)], the process of raining occurs using Eqs (9) and (10); otherwise, go to Step 4.

Step 7. Stop the algorithm if the termination condition is satisfied; otherwise, go to Step 4.

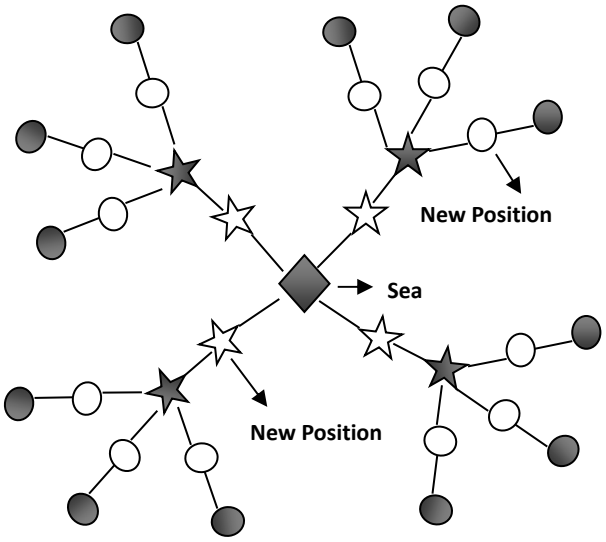

Figure 3. Schematic of a water cycle system.

3. Complex-valued encoding WCA

3.1. Complex-valued encoding method

In the CWCA, the imaginary and real parts of the complex number are updated respectively for each individual, which leads to inherent parallelism, improves the diversity of individuals, and enhances its exploitation and exploration capabilities, and it does not fall into the local optimum.

3.1.1. Initialize the CWCA population

In the WCA, first, an interval $[LB_k, UB_k], k = 1,2,..., N_{pop}$ is defined, and the N_{pop} complex modulus and phase angle are randomly generated using

$$\rho_k \in \left[0, \frac{LB_k - UB_k}{2}\right], \ k = 1,2,\ldots N_{pop} \tag{11}$$

$$\theta_k \in [-2\pi, 2\pi], \ k = 1,2,\ldots, N_{pop}. \tag{12}$$

The $2M$ complex number is obtained as follows:

$$X_{Rk} + iX_{Ik} = \rho_k \left(\cos\theta_k + i\sin\theta_k\right),\ \ k = 1,2,\ldots N_{pop} \tag{13}$$

Therefore, in the CWCA, the initial population represented by a matrix of streams of size $N_{pop} \times N$ is composed as follows:

$$Population = \begin{bmatrix} Sea \\ River_1 \\ River_2 \\ River_3 \\ \vdots \\ Stream_{Nsr+1} \\ Stream_{Nsr+2} \\ Stream_{Nsr+3} \\ \vdots \\ Stream_{N_{pop}} \end{bmatrix} = \begin{bmatrix} \left[x_{R(1,1)}, x_{I(1,1)}\right] & \left[x_{R(1,2)}, x_{I(1,2)}\right] & \left[x_{R(1,3)}, x_{I(1,3)}\right] & \cdots & \left[x_{R(1,N)}, x_{I(1,N)}\right] \\ \left[x_{R(2,1)}, x_{I(1,1)}\right] & \left[x_{R(2,2)}, x_{I(2,2)}\right] & x_{2,3} & \cdots & \left[x_{R(2,N)}, x_{I(2,N)}\right] \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \left[x_{R(N_{pop},1)}, x_{I(N_{pop},1)}\right] & \left[x_{R(N_{pop},2)}, x_{I(N_{pop},2)}\right] & \left[x_{R(N_{pop},3)}, x_{I(N_{pop},3)}\right] & \cdots & \left[x_{R(N_{pop},N)}, x_{I(N_{pop},N)}\right] \end{bmatrix}, \tag{14}$$

where N_{pop} denotes the complex modulus.

3.1.2. Updating method of the CWCA

In the CWCA, the new positions of rivers and streams are determined as follows: the real parts are updated using

$$X_{R\_Stream}^{k+1} = X_{R\_Stream}^{k} + rand \times C \times (X_{R\_River}^{k} - X_{R\_Stream}^{k}) \tag{15}$$

$$X_{R\_River}^{k+1} = X_{R\_River}^{k} + rand \times C \times (X_{R\_Sea} - X_{R\_River}^{k}), \tag{16}$$

and the imaginary parts are updated using

$$X_{I\_Stream}^{k+1} = X_{I\_Stream}^{k} + rand \times C \times (X_{I\_River}^{k} - X_{I\_Stream}^{k}) \tag{17}$$

$$X_{I\_River}^{k+1} = X_{I\_River}^{k} + rand \times C \times (X_{I\_Sea} - X_{I\_River}^{k}). \tag{18}$$

3.1.3. Fitness value

The fitness value is determined as follows: First, the complex value is converted into a real value and then its fitness value is calculated. The real-valued fitness function is determined using the complex modules and the sign is updated using the amplitude angle:

$$\rho_k = \sqrt{X_{Rk}^2 + X_{Ik}^2},\quad k = 1,2,\ldots,N_{pop} \tag{19}$$

$$X_k = \rho_k \operatorname{sgn}\left(\sin\left(\frac{X_{Ik}}{\rho_k}\right)\right) + \frac{UB_k + LB_k}{2},\quad k = 1,2,\ldots,N_{pop}, \tag{20}$$

where X_k denotes the converted real variables.

3.2. Pseudocode of the CWCA

The pseudocode of the CWCA is as follows:

Pseudocode of the CWCA

1. **BEGIN**
2. Initial population parameters: N_{pop}, N_{sr}, and let $\rho_k \in \left[0, \frac{LB_k - UB_k}{2}\right], \theta_k \in [-2\pi, 2\pi]$.
3. Convert the complex value into real variables [Eqs (19) and (20)]. Using Eqs (14), (2) and (3), randomly generate the initial population and form the initial rivers, streams, and ocean, respectively.
4. Evaluate the intensity of flow for rivers [Eq (4)].
5. **while** ($iter < iterMax$)
6: Calculate the real part using Eqs (14) and (15).
7. Calculate the imaginary part using Eqs (17) and (18).
8. Convert the complex value into real variables using Eqs (19) and (20).
   Swap the positions of the rivers, streams, and ocean with respect to their fitness values.
9. **for** each river **do**
10. **if** $\left|X_{sea} - X_{River}^{k}\right| < eps$
11. Rain began, and new streams and rivers are generated [Eqs (11–13)]
12. **end if**
13. **end for**
14. **end while**
15. Output the optimal solution
16. **END**

4. Experimental results and comparisons

The experimental setup was as follows: MATLAB R2012a, AMD Athlon™*4 640 processor, and 2 GB memory. The CWCA was used to solve 12 benchmark functions [18,19] and four engineering design problems. The performance of CWCA was compared with that of GSO [4], artificial bee colony (ABC) [20], WCA [11], and ERWCA [14] using the standard deviation and mean. The control parameters of the algorithms were set as follows:

- GSO: Parameters N_{pop} =50, $G_0 = 100, \alpha = 20; K_0$ linearly decreased to 1 and was set to NP [4].
- ABC: Parameters N_{pop} = 50, $Limit = 5D$. [20].
- WCA: Parameters N_{pop} =s 50, $N_{sr} = 8$ (number of rivers), with $C = 2$. [11].

- ERWCA: Parameters $N_{pop} = 50$, $N_{sr} = 8$, with $C = 2$. [14].
- CWCA: Parameters $N_{pop} = 50, N_{sr} = 8$, with $C = 2$, $\rho_k \in [0, \frac{LB_k - UB_k}{2}], \theta_k \in [-2\pi, 2\pi]$.

For the simulation experiments, the mean outcomes of 25 independent runs for $f_{01} - f_{04}$ are listed in Table 2, the test results for functions $f_{05} - f_{08}$ are listed in Table 3, and the test results for functions $f_{09} - f_{12}$ are listed in Table 4. The unrestrained benchmark functions are presented in Table 1. The results perform 25 generations. "Best," "Mean," "Worst," "Std," and "ANOVA" represent the optimal value, mean value, worst value, standard deviation, and analysis of variance, respectively.

Table 1. Test functions for the benchmark.

Unstrained functions	D	Range	Optimum	Iter
$f_{01} = \sum_{i=1}^{D} x_i^2$	50	[–100, 100]	0	500
$f_{02} = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	50	[–10, 10]	0	500
$f_{03} = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	50	[–100, 100]	0	500
$f_{04} = \max \{ \lvert x_i \rvert, 1 \le i \le D \}$	50	[–100, 100]	0	500
$f_{05} = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor \right)^2$	50	[–100, 100]	0	100
$f_{06} = \sum_{i=1}^{D} \left[x_i^2 - 10\cos 2\pi x_i + 10 \right]$	50	[–5.12, 5.12]	0	100
$f_{07} = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i) + 20 + e$	50	[–32, 32]	0	100
$f_{08} = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	50	[–600, 600]	0	100
$f_{09} = 4x_i^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[–5, 5]	–1.03162	100
$f_{10} = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2) \right] \times \left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2) \right]$	2	[–5, 5]	3	100
$f_{11} = -\sum_{i=1}^{4} c_i \exp\left[\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2 \right]$	3	[0, 1]	–3.8628	100
$f_{12} = -\sum_{i=1}^{4} c_i \exp\left[\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2 \right]$	6	[0, 1]	–3.3224	100

Table 2. Test results for the benchmark functions $f_{01} - f_{04}$.

Functions	Algorithms	Results			
		Best	Mean	Worst	Std.
$f_{01}(D=50)$	GSO	0.207355483	0.643658692	1.381211712	0.305186488
	ABC	0.02164161	0.266296242	1.0272904	0.308026332
	WCA	27.27537295	60.81289202	125.1674416	24.49799538
	ERWCA	3.65357E–16	2.23497E–13	1.2168E–12	2.98299E–13
	CWCA	**1.9495E–207**	7.9173E–188	1.9695E–186	0
$f_{02}(D=50)$	GSO	0.190471323	0.523626056	1.926347272	0.35741995
	ABC	0.083559555	0.159897637	0.284741881	0.054138322
	WCA	12.77983257	73286.99059	1666293.509	332935.3144
	ERWCA	5.40146E–09	0.000105465	0.002580972	0.000515735
	CWCA	**3.0115E–108**	4.8137E–101	5.05153E–100	1.29853E–100
$f_{03}(D=50)$	GSO	5614.494834	9162.038749	13171.78451	2191.964722
	ABC	49854.83613	63062.58662	80673.90543	7903.826327
	WCA	22036.10203	40160.29047	71405.33553	11778.719
	ERWCA	1.48038E–14	3.58088451	31.08218217	7.333461723
	CWCA	**1.0586E–175**	1.0027E–155	2.0513E–154	4.1614E–155
$f_{04}(D=50)$	GSO	7.482344366	11.02103367	15.22690812	1.805226231
	ABC	68.25830295	80.27564106	87.87728499	4.794442375
	WCA	48.67045644	66.08139632	81.93733451	7.664963523
	ERWCA	2.43589E–10	5.55482E–08	2.43961E–07	5.29924E–08
	CWCA	**7.15741E–99**	3.31169E–92	3.17129E–91	7.72028E–92

4.1. Experimental results for the benchmark functions

As shown in Table 2, the best values produced by the CWCA were more accurate than those obtained using the proposed WCA and the other meta-heuristic optimization algorithms. For four functions, the standard deviation of the CWCA was less than that of GSO, ABC, WCA, and ERWCA, which implies that the CWCA had better stability in terms of optimizing high-dimensional unimodal functions. Figures 4–7 show that the fitness functions converged to a curve. Hence, it can be concluded that the proposed CWCA had a higher convergence rate and higher computation precision

than meta-heuristic algorithms such as GSO, ABC, WCA, ERWCA, and CWCA. Figures 8–11 show the analysis of ANOVA using a graph. The ERWCA and CWCA obtained stable optimal values. Figure 9 show that most of the algorithms produced a stable value, except the WCA, when solving function f_{02}.

Table 3. Test results for the benchmark functions $f_{05} - f_{08}$.

Functions	Algorithms	Results			
		Best	Mean	Worst	Std.
$f_{05}(D=50)$	GSO	566	1131.64	1950	342.5811193
	ABC	13,559	27,666.12	44,007	7900.040123
	WCA	8099	13,872.64	24,084	4340.085031
	ERWCA	0	0.76	19	3.8
	CWCA	**0**	0	0	0
$f_{06}(D=50)$	GSO	102.9655468	130.6203352	186.189766	18.88401764
	ABC	219.8304665	322.5058351	377.4759954	40.06728218
	WCA	299.2085116	440.4852994	533.5915321	59.06933452
	ERWCA	0	1.989918114	49.74795285	9.949590571
	CWCA	**0**	0	0	0
$f_{07}(D=50)$	GSO	5.764211343	8.469214323	10.36140998	1.216196283
	ABC	16.85525103	18.18606241	18.93602973	0.470872533
	WCA	5.598512115	16.86361644	20.34625814	4.444925239
	ERWCA	2.84823E–09	1.74804E–05	0.000429133	8.57614E–05
	CWCA	**8.88178E–16**	8.88178E–16	8.88178E–16	0
$f_{08}(D=50)$	GSO	1.11022E–16	0.052857507	0.073410582	0.033637865
	ABC	3.33067E–16	0.003443108	0.073410582	0.014668708
	WCA	6.24611E–13	0.063455934	0.195437086	0.047329549
	ERWCA	0	5.41167E–14	8.64864E–13	1.70589E–13
	CWCA	**0**	0	0	0

It can be noted from Table 3 that the CWCA located the optimal solutions of the functions f_{05}, f_{06} and f_{08} with a standard deviation of 0. For function f_{07}, the precision and mean value were higher than those for the other meta-heuristic optimization algorithms. Building on the results shown in Figures 12–15, it can be concluded that the CWCA converged faster and its precision was higher than that of the other meta-heuristic optimization algorithms. Figures 16–19 show the ANOVA graphical analysis results. The results demonstrated that both the ERWCA and CWCA had better stability than the other meta-heuristic optimization algorithms.

Table 4. Test results for the benchmark functions $f_{09} - f_{12}$.

Functions	Algorithms	Results			
		Best	Mean	Worst	Std.
$f_{09}(D=2)$	GSO	**–1.031628453**	–1.031628453	–1.031628453	4.2276E–16
	ABC	–1.031628453	–1.031628453	–1.031628453	4.03633E–12
	WCA	–1.031628453	–1.031628453	–1.031628453	1.16441E–10
	ERWCA	–1.031628453	–1.031628453	–1.031628453	6.06115E–14
	CWCA	–1.031628453	–1.031628453	–1.031628453	8.87086E–11
$f_{10}(D=2)$	GSO	**3**	3	3	4.39868E–13
	ABC	3	6.829312618	34.11721073	9.461128153
	WCA	3	3.000000007	3.000000081	1.62655E–08
	ERWCA	3	3	3	7.70669E–12
	CWCA	3	3.000000005	3.000000093	1.8483E–08
$f_{11}(D=3)$	GSO	–3.862782148	–3.862781304	–3.86277018	2.56717E–06
	ABC	–3.862782148	–3.831845002	–3.0897641	0.154600208
	WCA	–3.862782148	–3.862782122	–3.862781721	8.52233E–08
	ERWCA	–3.862782148	–3.862782148	–3.862782148	1.67437E–11
	CWCA	**–3.862782148**	–3.862782148	–3.862782147	7.21169E–11
$f_{12}(D=6)$	GSO	–3.322368009	–3.274490605	–3.199245153	0.059836344
	ABC	–3.322367256	–3.322206338	–3.319008561	0.000668069
	WCA	–3.322367865	–3.283354564	–3.186190502	0.058086807
	ERWCA	–3.322368011	–3.312831524	–3.203161909	0.033006674
	CWCA	**–3.322368011**	–3.322367989	–3.322367817	4.5094E–08

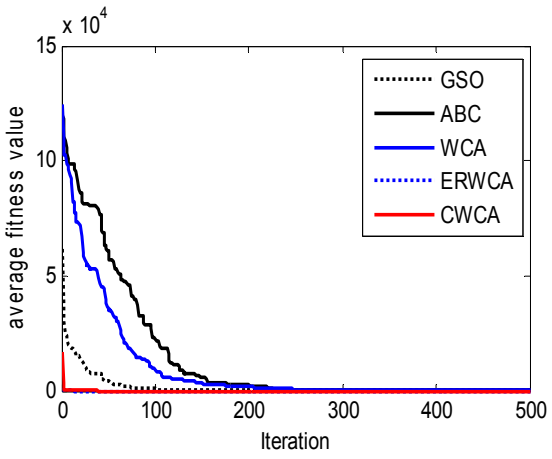

Figure 4. f_{01} function evolution curve.

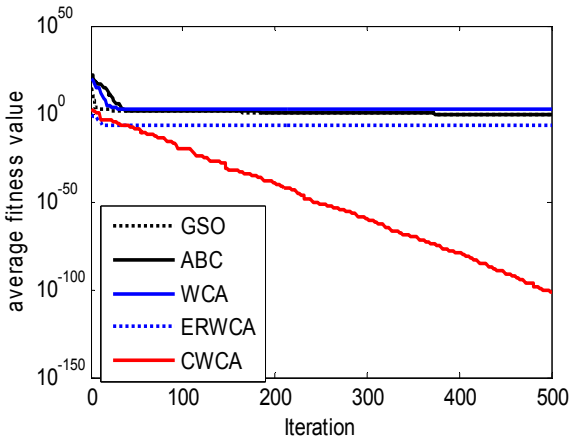

Figure 5. f_{02} function evolution curve.

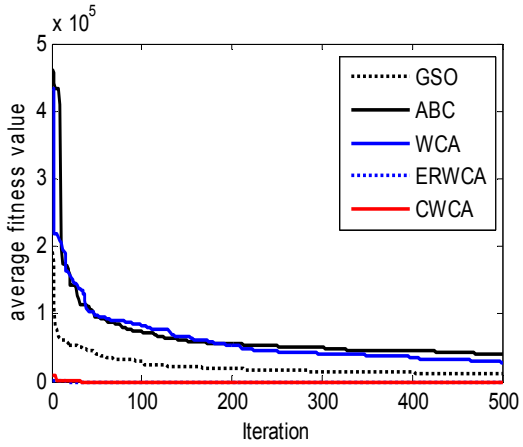

Figure 6. f_{03} function evolution curve.

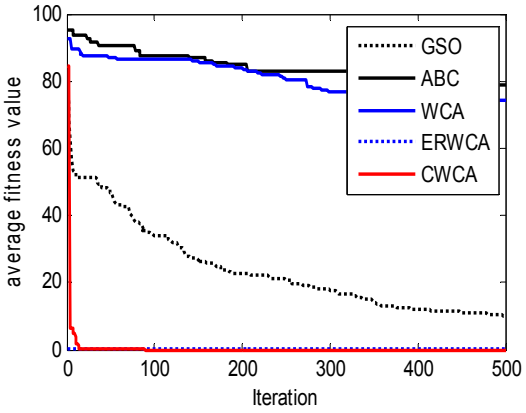

Figure 7. f_{04} function evolution curve.

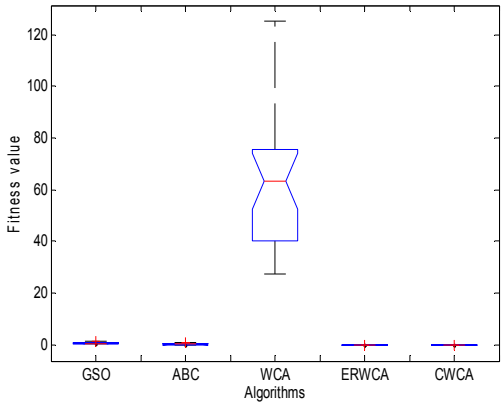

Figure 8. ANOVA for function f_{01}.

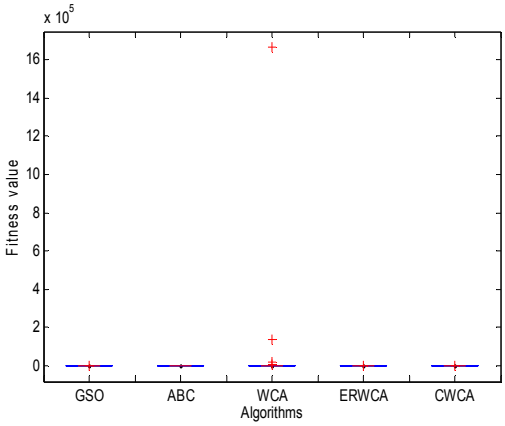

Figure 9. ANOVA for function f_{02}.

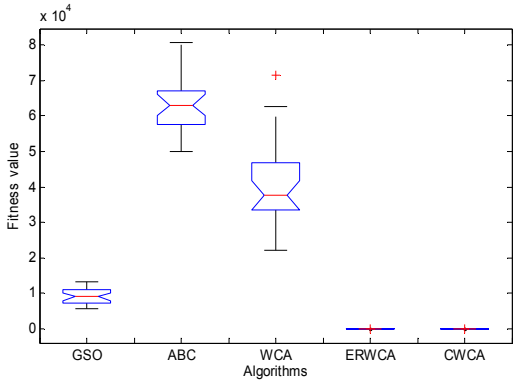

Figure 10. ANOVA for function f_{03}.

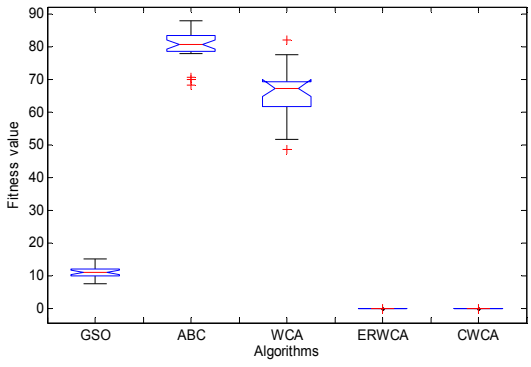

Figure 11. ANOVA for function f_{04}.

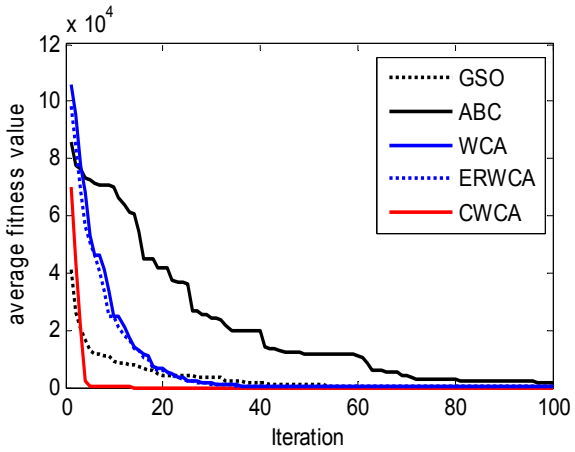

Figure 12. Fitness function evolution curve for f_{05}.

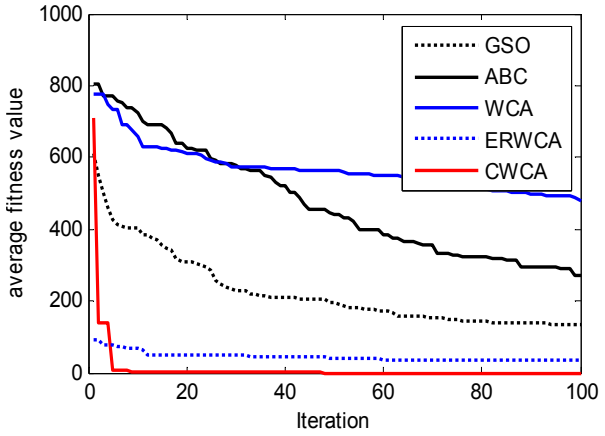

Figure 13. Fitness function evolution curve for f_{06}.

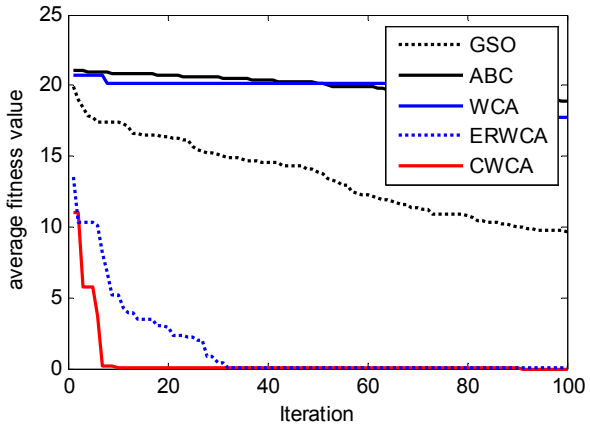

Figure 14. Fitness function evolution curve for f_{07}.

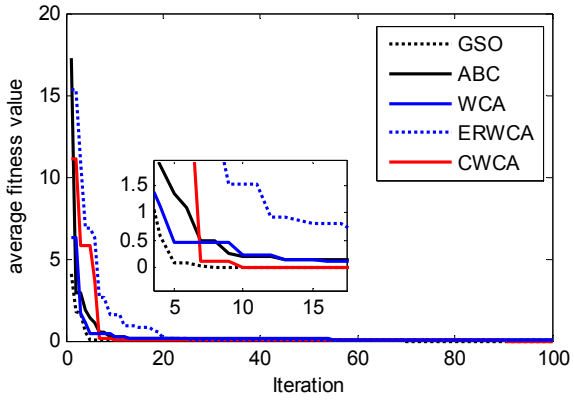

Figure 15. Fitness function evolution curve for f_{08}.

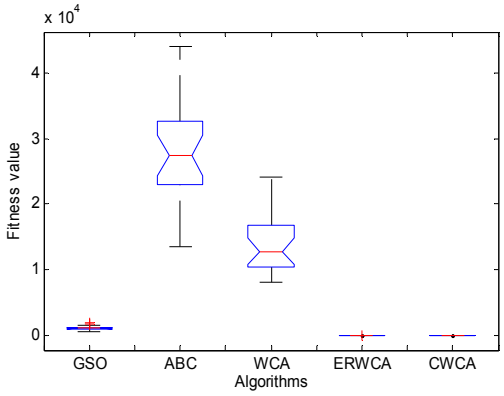

Figure 16. ANOVA for function f_{05}.

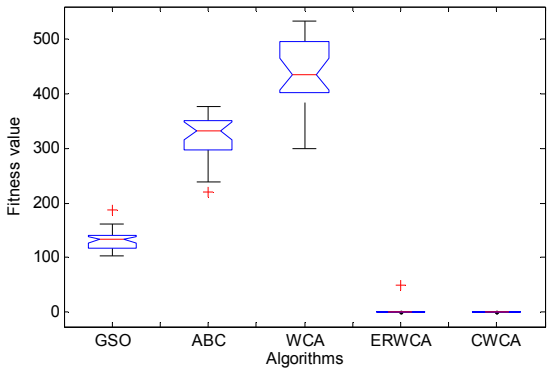

Figure 17. ANOVA for function f_{06}.

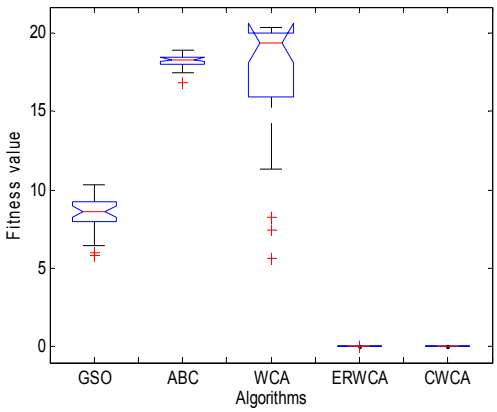

Figure 18. ANOVA for function f_{07}.

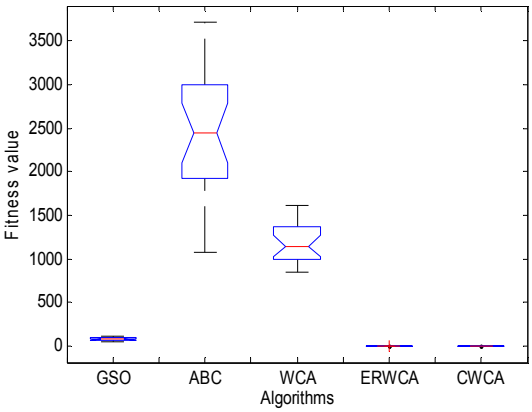

Figure 19. ANOVA for function f_{08}.

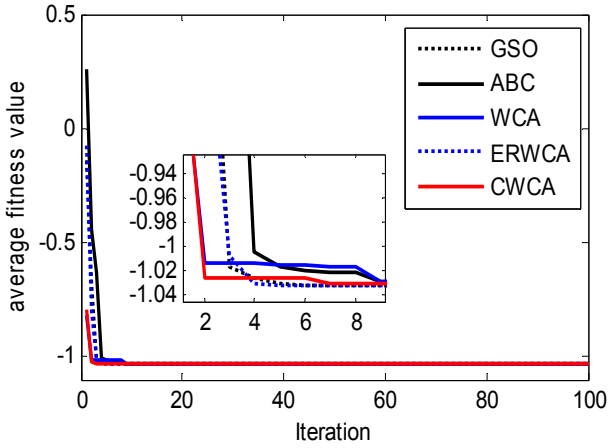

Figure 20. Fitness function evolution curve for f_{09}.

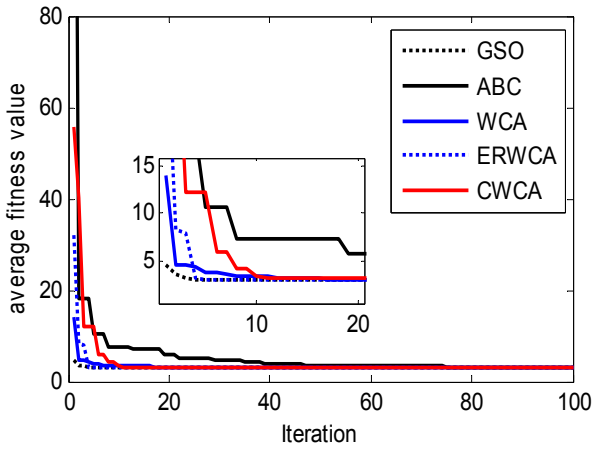

Figure 21. Fitness function evolution curve for f_{10}.

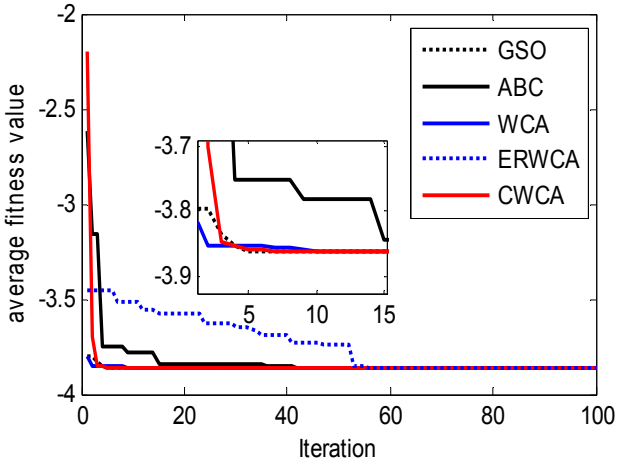

Figure 22. Fitness function evolution curve for f_{11}.

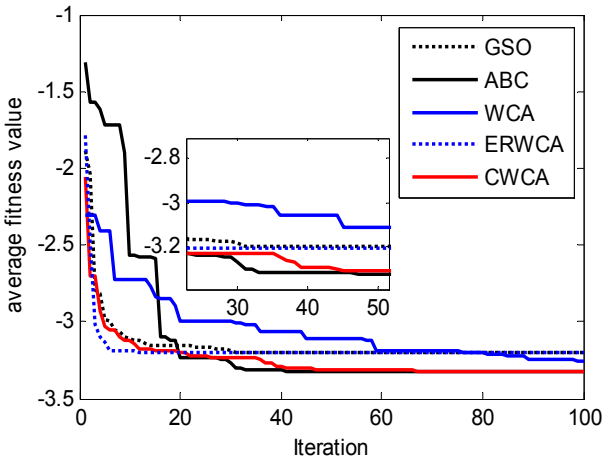

Figure 23. Fitness function evolution curve for f_{12}.

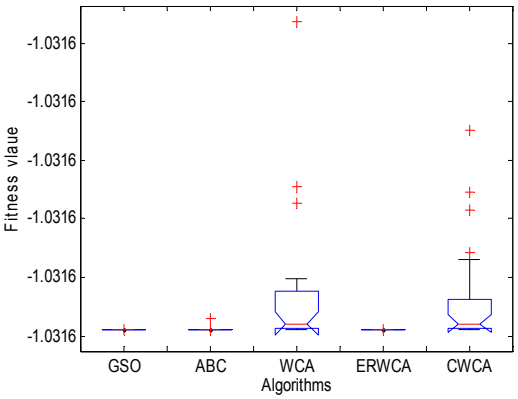

Figure 24. ANOVA for function f_{09}.

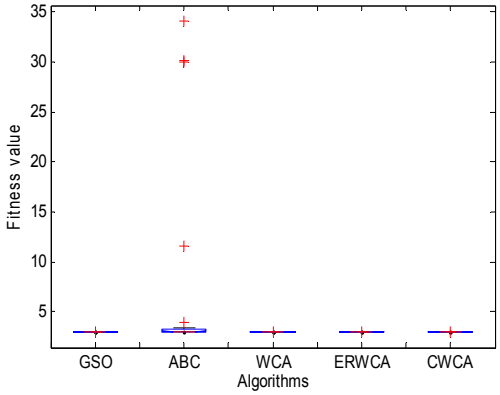

Figure 25. ANOVA for function f_{10}.

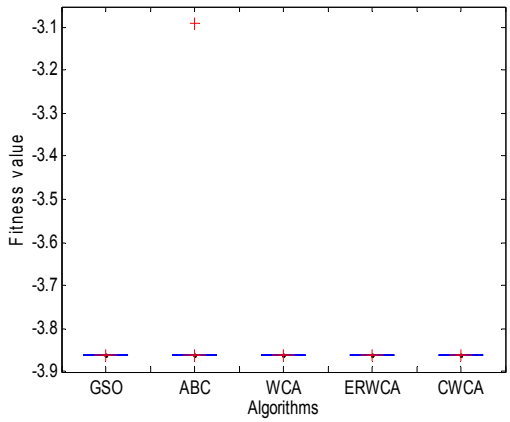

Figure 26. ANOVA for function f_{11}.

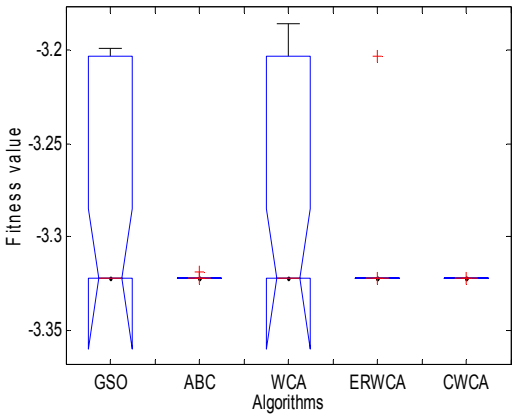

Figure 27. ANOVA for function f_{12}.

As shown in Table 4, for f_{09}, all algorithms produced optimal solutions, and the solutions of GSO were the most stable. For functions f_{10} and f_{11}, the solutions for most algorithms achieved the same order of magnitude, but the standard deviation of the CWCA was the best. For function f_{12}, both the ERWCA and CWCA obtained optimal solutions, but the CWCA had the best stability. Figures 20–27 shows that the CWCA had excellent performance when solving multimodal low-dimensional problems.

4.2. Engineering design problem

The mathematical model for general constrained optimization is defined [21] as

$$\begin{aligned} &Minimize \quad f(x); \\ &subject \ \ to \quad g_j(x) \le 0, \quad j = 1,\ldots,q, \\ &\qquad\qquad h_i(x) = 0, \quad i = q+1,\ldots,m, \\ &\qquad\qquad l_i \le x_i \le u_i, \quad i = 1,\ldots,D, \end{aligned} \tag{21}$$

where $x = (x_1, x_2, ..., x_D)$ represents the D-dimensional decision variables, $f(x)$ denotes the objective function, $g_j(x) \le 0$ represents q inequality constraints, and $h_i(x)$ represents $m-q$ equality constraints. The functions f, g_j and h_i are nonlinear or linear functions. l_i, u_i denote the lower and upper bounds, respectively.

Generally, when practical problems are solved, it is common to convert equality constraints into inequality constraints:

$$\left|h_i(x)\right| - \varepsilon \le 0, \quad i = q+1,\ldots,m\ , \tag{22}$$

where ε denotes the allowed tolerance, which is set to a small value. The best numerical value for ε is 0.001. According to the feasibility-based rules, each individual's constraint violation is

$$f_{viol}(x) = \max(f_{viol}^1, f_{viol}^2) \tag{23}$$

$$f_{viol}^1 = \max(g_j(x)) \qquad j = 1,\ldots,q \tag{24}$$

$$f_{viol}^2 = \max(\left|h_i(x)\right| - \varepsilon) \qquad i = q+1,\ldots,m\ . \tag{25}$$

Generally, a feasibility-based rule is used [22]:

Rule 1: Any feasible solution is preferred over any infeasible solution.

Rule 2: The feasible solution that has a better objective function value is preferred, if choosing between two feasible solutions.

Rule 3: The infeasible solution that has smaller constraint violations is preferred, if choosing between two infeasible solutions.

4.2.1. Pressure vessel design problem

A cylindrical pressure vessel with hemispherical heads capped at both ends is shown in Figure 28, where $T_s(x_1)$ denotes the shell thickness, $T_h(x_2)$ denotes the head thickness, $R(x_3)$ denotes

the inner radius, and $L(x_4)$ denotes the cylindrical vessel length. The overall cost, including a combination of material cost, single 60^0 welding cost, and forming cost, is minimized.

The mathematical model for the pressure vessel design problem is

$$\begin{aligned} &\textit{Minimize} \quad f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661xx + 19.84x_1^2x_4 \\ &\textit{Subject to} \quad g_1(x) = -x_1 + 0.0193x_3 \le 0 \\ &\qquad g_2(x) = -x_2 + 0.00954x_3 \le 0 \\ &\qquad g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0 \\ &\qquad g_4(x) = x_4 - 240 \le 0 \\ &\qquad 0 \le x_1, x_2 \le 100 \\ &\qquad 10 \le x_3, x_4 \le 200 \end{aligned} \tag{26}$$

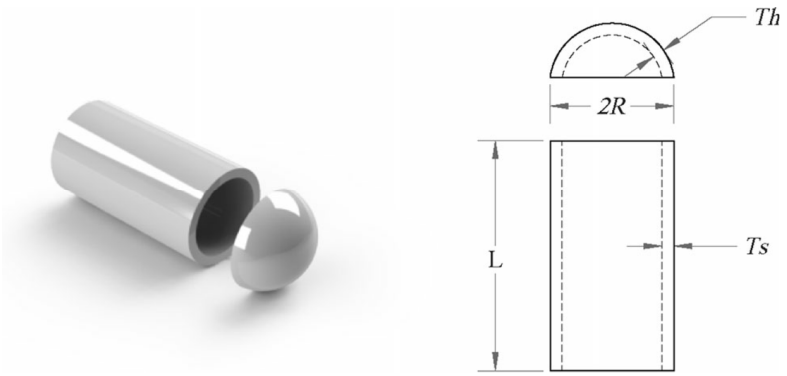

Figure 28. Schematic of the pressure vessel design problem [23].

Table 5. Comparison of the statistical results for the pressure vessel design problem.

Algorithms	Optimal values for variables				Optimum cost
	T_s	T_h	R	L	
ES [27]	0.8125	0.4375	42.098087	176.640518	6059.7456
GSA [28]	1.1250	0.6250	55.9886598	84.4542025	8538.8359
PSO [29]	0.8125	0.4375	42.091266	176.746500	6061.0777
GA [21]	0.8125	0.4375	40.323900	200.000000	6288.7445
GA [22]	0.8125	0.4375	42.097398	176.654050	6059.9463
GA [23]	0.9375	0.5000	48.329000	112.679000	6410.3811
DE [30]	0.8125	0.4375	42.098411	176.637690	6059.7340
ACO [31]	0.8125	0.4375	42.103624	176.572656	6059.0888
MVO [23]	0.8125	0.4375	42.0907382	176.738690	6060.8066
WCA [11]	1.7958	0.3933	41.2310	187.6901	5916.0914
CWCA	0.7782	0.3846	40.3196	200.0000	**5885.3327**

The results obtained using the CWCA to solve the pressure vessel design problem is shown in Table 5. The statistical outcomes were compared with those of the meta-heuristic algorithms ES [27], GSA [28], PSO [29], GA [24–26], DE [30], MVO [23], WCA [11], and ACO [31]. As shown in Table 5, the proposed CWCA outperformed all the other optimization algorithms.

4.2.2. Cantilever beam design problem

Cantilever beam design optimization aims to minimize the weight of a cantilever beam, which is composed of hollow square blocks. In this study, it involved five squares: the first block was fixed, and the fifth block had the burden of a vertical load. In Figure 29, five parameters define the shape of the cross section of the cubes.

The mathematical model for the cantilever beam design optimization is

$$\begin{aligned} &\textit{Minimize} \quad f(x) = 0.06224(x_1 + x_2 + x_3 + x_4 + x_5) \\ &\textit{Subject to} \quad g(x) = \frac{61}{x_1^3} + \frac{27}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0 \\ &\qquad\qquad 0.01 \le x_1, x_2 x_3, x_4, \le 100 \end{aligned} \tag{27}$$

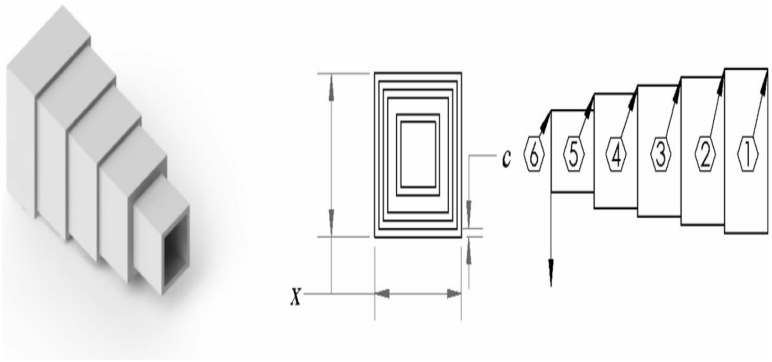

Figure 29. Schematic of the cantilever beam design problem [23].

Table 6. Statistical comparison results for the cantilever beam design problem.

Algorithms	Optimal variable values					Optimum
	x_1	x_2	x_3	x_4	x_5	weight
CS [32]	6.0089	5.3049	4.5023	3.5077	2.1504	1.3340
SOS [33]	6.01878	5.3034	4.4958	3.4989	2.1556	1.3340
MMA [34]	6.0100	5.3000	4.4900	3.4900	2.1500	1.3400
MVO [23]	6.0239	5.3060	4.49501	3.4960	2.1527	1.3399
WCA [11]	5.9799	4.8821	4.4659	3.4733	2.1380	1.30325382
CWCA	5.9783	4.8762	4.4663	3.4791	2.1391	**1.30325143**

In the same way, the meta-heuristic optimization algorithms MVO [23], CS [32], SOS [33], MMA [34], WCA [11], and CWCA were used. Table 6 shows that the results of the CWCA for the cantilever beam design problem were consistent with those of the other engineering optimization problems. Both the WCA and CWCA outperformed all the other optimization algorithms.

4.2.3. Three-bar truss design problem

The design problem for a three-bar planar truss structure is shown in Figure 30. The volume of a statically loaded three-bar truss was minimized, subject to stress (σ) constraints. The objective was to find the best cross-sectional areas (A_1, A_2).

The mathematical model for the three-bar planar truss structure design problem is

$$\begin{aligned} &Minimize \quad f(x) = (2\sqrt{2}x_1 + x_2) \times l \\ &Subject\ to \quad g_1(x) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0 \\ &\qquad g_2(x) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x} P - \sigma \le 0 \\ &\qquad g_3(x) = \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \le 0 \\ &\qquad 0 \le x_1, x_2 \le 1 \\ &\qquad l = 100cm, \quad P = 2kn/cm^2, \quad \sigma = kn/cm^2, \quad x_1 = A_1, \ x_2 = A_2 \quad A_1 = A_3 \end{aligned} \tag{28}$$

Table 7. Comparison of the statistical results for the three-bar truss design problem.

Algorithms	Optimal values for variables		Optimal weight
	A_1	A_2	
DEDS [34]	0.78867513	0.40824828	**263.8958434**
PSO-DE[35]	0.7886751	0.4082482	263.8958433
MBA [36]	0.7885650	0.4085597	263.8958522
CS [37]	0.78867	0.40902	263.9716
MVO [23]	0.78860276	0.40845307	263.8958499
WCA [11]	0.7888	0.4080	263.89585025
CWCA	0.7887	0.4082	**263.89584340**

For the three-bar truss design problem, the metaheuristic optimization algorithms MVO [23], DEDS [34], PSO-DE [35], MBA [36], CS [37], WCA [11], and CWCA were used. The proposed CWCA was compared with the other optimization algorithms (see Table 7). The results obtained using the CWCA were very close to those obtained using DEDS.

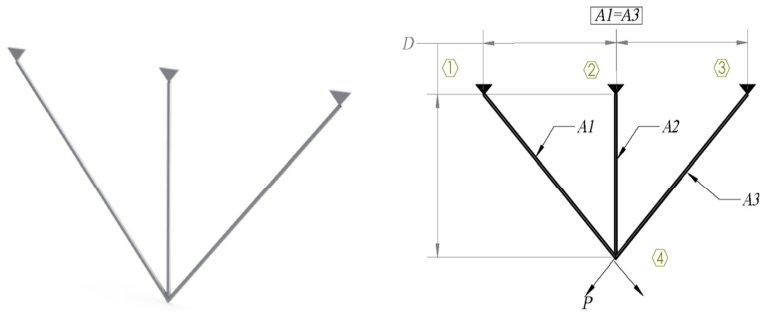

Figure 30. Schematic of the three-bar truss design problem [23].

4.2.4. Welded beam design problem

Figure 31 shows a beam made of low-carbon steel and welded to a rigid support. The welded beam was designed to achieve the minimum cost, subject to constraints on shear stress (τ), bending stress (σ) in the beam, buckling load on the bar (P_b), end deflection of the beam (δ), and side constraints. There were four design variables: $h(x_1), l(x_2), t(x_3)$ and $b(x_4)$.

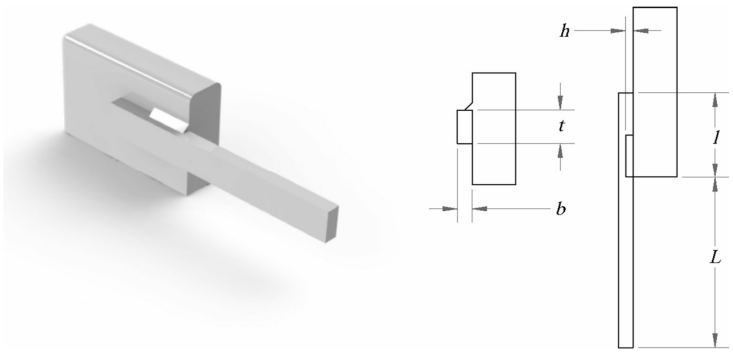

Figure 31. Schematic of the welded beam design problem [23].

The objective function is expressed as

$$
\begin{aligned}
&\textit{Minimize} \quad f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)\\
&\textit{Subject to} \quad g_1(x) = \tau(x) - \tau_{\max} \le 0\\
&\qquad\qquad\quad g_2(x) = \sigma(x) - \sigma_{\max} \le 0\\
&\qquad\qquad\quad g_3(x) = x_1 - x_4 \le 0\\
&\qquad\qquad\quad g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \le 0\\
&\qquad\qquad\quad g_5(x) = 0.125 - x_1 \le 0\\
&\qquad\qquad\quad g_6(x) = \delta(x) - \delta_{\max} \le 0\\
&\qquad\qquad\quad g_7(x) = P - P_c(x) \le 0\\
&\qquad\qquad\quad 0.1 \le x_1, x_4 \le 2 \qquad x_1 = h \quad x_2 = l\\
&\qquad\qquad\quad 0.1 \le x_2, x_3 \le 10 \qquad x_3 = t \quad x_4 = b
\end{aligned}
\tag{29}
$$

where

$$\tau(x)=\sqrt{(\tau')^2+2\tau'\tau''\frac{x_2}{2R}+(\tau'')^2}\quad \tau'=\frac{P}{\sqrt{2}x_1x_2}\quad \tau''=\frac{MR}{J}$$
$$M=P\left(L+\frac{x_2}{2}\right),\quad R=\sqrt{\frac{x_2^2}{4}+\left(\frac{x_1+x_3}{2}\right)^2},$$
$$J=2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12}+\left(\frac{x_1+x_3}{2}\right)^2\right]\right\} \tag{30}$$
$$\sigma(x)=\frac{6PL}{x_4x_3^2},\quad \delta(x)=\frac{4PL^3}{Ex_3^3x_4},\quad P_c(x)=\frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1-\frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$
$$P=6000lb,\quad L=14in,\quad E=30\times10^6\,psi,\quad G=12\times10^6\,psi$$
$$\tau_{\max}=13600\,psi,\quad \sigma_{\max}=30000\,psi,\quad \delta_{\max}=0.25in$$

Table 8. Comparison of the statistical results for the welded beam design problem.

Algorithms	Optimal values for variables				Optimal
	h	*l*	*t*	*b*	cost
GSA [28]	0.1821	3.856979	10.0000	0.202376	1.87995
CPSO [38]	0.2023	3.544214	9.048120	0.205723	1.72802
GA [39]	0.2489	6.1730	8.1789	0.2533	2.4331
HS [40]	0.2442	6.2231	8.2915	0.2443	2.3807
MVO [23]	0.2054	3.473193	9.044502	0.205695	1.72645
WCA [11]	0.2058	3.4697	9.0353	0.2058	1.7252
CWCA	0.2057	3.4705	9.0366	0.2057	**1.7248**

In the same way, the meta-heuristic optimization algorithms MVO [23], GSA [28], CPSO [38], GA [39], and HS [40] were applied. The proposed CWCA was compared with the other meta-heuristic algorithms (see Table 8). The results showed that the proposed CWCA achieved the minimum cost.

5. Conclusions and future directions

In recent years, significant attention has been paid to the design of meta-heuristic optimization algorithms to solve optimization problems. The aims of this study were to improve the exploration and exploitation capabilities of the WCA algorithm, accelerate its convergence speed, and enhance its calculation accuracy. A novel CWCA algorithm was proposed, and to evaluate the performance of CWCA optimization, 12 benchmark functions and four engineering examples were considered. The test results indicated that the CWCA had higher precision and convergence speed than the WCA and other popular meta-heuristic optimization algorithms. Further improvements to versions of the WCA equipped with the latest, efficient strategies should be considered in future research. Additionally, the

CWCA is not well used in some engineering majors, such as mechanical engineering. Despite the great potential of this principal field, it is necessary to use and use more meta-heuristic optimization algorithms, such as CWCA, to solve large-scale optimization problems.

Acknowledgments

This work was supported by the National Science Foundation of China under Grant No. 62066005 and the Project of Guangxi Natural Science Foundation under Grant No. 2018GXNSFAA138146. We thank Maxine Garcia, Ph.D, from Liwen Bianji (Edanz) (www.liwenbianji.cn/) for editing the English text of a draft of this manuscript.

Conflict of interest

The authors declare no conflict of interest.

References

1. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multi-objective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.*, **6** (2002), 182–197.
2. J. Kennedy, Particle swarm optimization, in *Encyclopedia of Machine Learning*, Springer US, (2010), 760–766.
3. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, *Science*, **220** (1983), 671–680.
4. K. N. Krishnanand, D. Ghose, Glowworm swarm optimisation: A new method for optimising multi-modal functions, *Int. J. Comput. Intell. Stud.*, **1** (2009), 93–119.
5. B. Alatas, Chaotic harmony search algorithms, *Appl. Math. Comput.*, **216** (2010), 2687–2699.
6. K. M. Passino, Biomimicry of bacterial foraging for distributed optimization and control, *IEEE Control Syst. Mag.*, **22** (2002), 52–67.
7. G. G. Wang, S. Deb, L. D. S. Coelho, Earthworm optimization algorithm: a bio-inspired metaheuristic algorithm for global optimization problems, *J. Bio-Inspired Comput.*, **12** (2018), 1–22.
8. X. S. Yang, A new metaheuristic bat-inspired algorithm, in *Nature inspired cooperative strategies for optimization*, Springer Berlin Heidelberg, (2010), 65–74.
9. G. G. Wang, S. Deb, X. Z. Gao, L. D. S. Coelho, A new metaheuristic optimization algorithm motivated by elephant herding behavior, *J. Bio-Inspired Comput.*, **8** (2017), 394–409.
10. G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, J. Li, Incorporating mutation scheme into krill herd algorithm for global numerical optimization, *Neural Comput. Appl.*, **24** (2014), 853–871.
11. H. Eskandar, A. Sadollah, A. Bahreininejad, M. Hamdi, Water cycle algorithm–A novel metaheuristic optimization method for solving constrained engineering optimization problems, *Comput. Struct.,* **110** (2012), 151–166.
12. A. Sadollah, H. Eskandar, A. Bahreininejad, J. H. Kim, Water cycle algorithm for solving multi-objective optimization problems, *Soft Comput.*, **19** (2015), 2587–2603.
13. C. Zhang, G. W. Liao, L. Li, Optimizations of space truss structures using WCA algorithm, *Prog. Steel Build. Struct.*, **1** (2014), 35–38.

14. A. Sadollah, H. Eskandar, A. Bahreininejad, J. H. Kim, Water cycle algorithm with evaporation rate for solving constrained and unconstrained optimization problems, *Appl. Soft Comput.*, **30** (2015), 58–71.
15. L. Li, Y. Zhou, A novel complex-valued bat algorithm, *Neural Comput. Appl.*, **25** (2014), 1369–1381.
16. D. B. Chen, H. J. Li, Z. Li, Particle swarm optimization based on complex-valued encoding and application in function optimization, *Comput. Appl.*, **45** (2009), 59–61.
17. Z. Zheng, Y. Zhang, Y. Qiu, Genetic algorithm based on complex-valued encoding, *Control Theory Appl.*, **20** (2003), 97–100.
18. X. S. Yang, Appendix A: test problems in optimization, *Eng. Optim.*, **2010** (2010), 261–266.
19. K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, et al., Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. *Nat. Inspired Comput. Appl. Lab.*, **2007** (2007), 153–177.
20. D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *J. Global Optim.*, **39** (2007), 459–471.
21. C. A. C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, *Comput. Methods Appl. Mech. Eng.*, **191** (2002)**,** 1245–1287.
22. E. Mezura-Montes, C. A. C. Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, *Int. J. Gen. Syst.*, **37** (2008), 443–473.
23. S. Mirjalili, S. M. Mirjalili, A. Hatamlou, Multi-Verse Optimizer: a nature-inspired algorithm for global optimization, *Neural Comput. Appl.*, **27** (2016), 495–513.
24. C. A. C. Coello, Use of a self-adaptive penalty approach for engineering optimization problems, *Comput. Ind.*, **41** (2000), 113–127.
25. C. A. C. Coello, Montes, E. M. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, *Adv. Eng. Inf.*, **16** (2002),193–203.
26. K. Deb, Geneas: A robust optimal design technique for mechanical component design, in *Evolutionary algorithms in engineering applications*, Springer Berlin Heidelberg, (1997), 497–514.
27. E. Mezura-Montes, C. A. C. Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, *Int. J. Gen. Syst.*, **37** (2008), 443–473.
28. E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: A gravitational search algorithm, *Inf. Sci.*, **179** (2009), 2232–2248.
29. Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, *Eng. Appl. Artif. Intell.*, **20** (2007), 89–99.
30. L. J. Li, Z. B. Huang, F. Liu, Q. H. Wu, A heuristic particle swarm optimizer for optimization of pin connected structures, *Comput. Struct.*, **85** (2007), 340–349.
31. A. Kaveh, S. Talatahari, An improved ant colony optimization for constrained engineering design problems, *Eng. Comput.*, **27** (2010), 155–182.
32. A. H. Gandomi, X. S. Yang, A. H. Alavi, Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems, *Eng. Comput.*, **29** (2013), 17–35.
33. M. Y. Cheng, D. Prayogo, Symbiotic Organisms Search: A new metaheuristic optimization algorithm, *Comput. Struct.*, **139** (2014), 98–112.

34. M. Zhang, W. Luo, X. Wang, Differential evolution with dynamic stochastic selection for constrained optimization, *Inf. Sci.*, **178** (2008), 3043–3074.
35. H. Liu, Z. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, *Appl. Soft Comput.*, **10** (2010), 629–640.
36. A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems, *Appl. Soft Comput.*, **13** (2013), 2592–2612.
37. A. H. Gandomi, X. S. Yang, A. H. Alavi, Cuckoo search algorithm: A metaheuristic approach tosolve structural optimization problems, *Eng. Comput.*, **29** (2013), 17–35.
38. R. Krohling, L. dos Santos Coelho, Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems, *IEEE Trans. Syst. Man Cyber. B Cyber.*, **36** (2006), 1407–1416.
39. K. Deb, An efficient constraint handling method for genetic algorithms, *Comput. Methods Appl. Mech. Eng.*, **186** (2000), 311–338.
40. K. S. Lee, Z. W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, *Comput. Methods Appl. Mech. Eng.*, **194** (2005)**,** 3902–3933.
41. N. Mohammad, S. Ali, H. C. Young, H. K. Joong, A comprehensive review on water cycle algorithm and its applications, *Neural Comput. Appl.*, **32** (2020), 7433–17488.
42. A. Sadollah, H. Eskandar, H. M. Lee, D. G. Yoo, J. H. Kim, Water cycle algorithm: A detailed standard code, *Softwarex*, **5** (2016), 37–43.
43. E. Osaba, J. Del Ser, A. Sadollah, M. N. Bilbao, D. Camacho, A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem, *Appl. Soft Comput.*, **71** (2018), 277–290.
44. M. Seyed, P. Abedi, A. Alireza, S. Ali, H. Joong, Gradient-based water cycle algorithm with evaporation rate applied to chaos suppression, *Appl. Soft Comput.*, **53** (2017), 420–440.
45. G. G. Wang, Y. Tan, Improving metaheuristic algorithms with information feedback models, *IEEE Trans. Cybern.*, **49** (2019), 542–555.
46. G. G. Wang, L. Guo, A. H. Gandomi, G. S. Hao, H. Wang, Chaotic krill herd algorithm, *Inf. Sci.*, **274** (2014), 17–34.
47. W. Deng, J. Xu, X. Z. Gao, H. Zhao, An enhanced MSIQDE algorithm with novel multiple strategies for global optimization problems, *IEEE Trans. Syst. Man Cybern. Syst.*, **2020** (2020).
48. Y. Jiang, Q. Luo, Y. Wei, L. Abualigah, An efficient binary Gradient-based optimizer for feature selection, *Math. Biosci. Eng.*, **18** (2021), 3813–3854.

Appendix

CWCA **MATLAB** source code

```
%----------------------------------------------
function [Xmin, Fmin] = CWCA(objective_function, LB, UB, nvars, Fnum, Nmax)
    nvar = nvars;       % Dimension
    N = 50;              % Population size
    Nsr = 8;             % Number of rivers
    C = 2;
```

```
    Nwat = N-Nsr;      % Raindrops
    Fmin = inf;
%---------- Initialization (In module and Angle mode)---------------------------------
    zigma_NCn = inf*ones(1, Nwat + 1);
for i=1: N
    x_age(i, :) = 4*pi*(rand(1, nvar)-0.5);            % Argument
    x_mod(i, :) = 0.5*(UB - LB).*rand(1, nvar);    % Module
    x_R(i,:) = x_mod(i, :).*cos(x_age(i,:));
    x_I(i, :) = x_mod(i,:).*sin(x_age(i,:));
    sq_x(i, :) = sqrt(x_R(i, :).^2 + x_I(i, :).^2);
    x(i, :) = sign(sin(x_I(i, :)/sq_x(i, :)))*sq_x(i, :) + 0.5*(UB + LB);
end
    LB_bound = repmat(LB, N, 1);
    UB_bound = repmat(UB, N, 1);
    x =max(x, LB_bound);
    x = min(x, UB_bound);
for i =1 : N
    xx(i, :) =    objective_function(x(i, :), Fnum);
end
    [~, index1] = sort(xx, 'ascend');
    Sr = x(index1(1: Nsr+1),:);
    sr_R=x_R(index1(1: Nsr+1), :);
    sr_I = x_I(index1(1: Nsr+1), :);
    SR = sr;
    SR_R = sr_R;
    SR_I = sr_I;
%-------------------------------------------------------------------------
for i = 1:Nsr+1
    cost(i) = objective_function(SR(i,:), Fnum);    % Calculate fitness values
end
    cs = sort(cost, 'ascend')';
    cs = cs(1: Nsr+1);
    CN = cs-max(cs);
    Pn = abs(CN/(sum(CN)));
    Pn(Nsr+1) = [ ];
    NCn = round(Nwat*Pn);
while (sum(NCn) ~= Nwat)
    i = Nsr;
while sum(NCn) > Nwat
    if NCn(i) <= 1
        i = i-1;
end
        NCn(i) = NCn(i)-1;
end
```

```
            i = 1;
while sum(NCn) < Nwat
            NCn(i) = NCn(i)+1;
end
end
      NCn = sort(NCn, 'descend');
      Sr = sr(1: Nsr, :);
      sr_R = sr_R(1: Nsr, :);
      sr_I=sr_I(1: Nsr, :);
      zigma_NCn = [0];
for i = 1:Nsr
      zigma_NCn = [zigma_NCn sum(NCn(1: i))];
end
end
%------------ Initialization ---------------------------------
for j = 1:Nwat
      water_age(j, :) = 4*pi*(rand(1, nvar)-0.5);
      water_mod(j, :) = 0.5*(UB - LB).*rand(1, nvar);
      water_R(j, :) = water_mod(j, :).*cos(water_age(j, :));
      water_I(j, :) = water_mod(j, :).*sin(water_age(j,:));
      sq_water = sqrt(water_R(j, :).^2 + water_I(j, :).^2);
      water(j, :) = sign(sin(water_I(j, :)/sq_water))*sq_water + 0.5*(UB + LB);
end
      WATER_R = water_R;
      WATER_I = water_I;
      WATER = water;
%-----------------------------------------------------
      Sea = sr(1,:);
      sea_R = sr_R(1, :);
      sea_I = sr_I(1, :);
%--------------- The main loop -----------------------------
      Locate = 1;
for ii = 1:Nmax      % Maximum number of iterations
%--------------- The stream flows to the river -------------------------
      stp = 1;
for i = 1: Nsr_k
      for j=(zigma_NCn(i) + 1): zigma_NCn(i+1)
%----------- Real component update --------
      new_WATER_R= WATER_R(j, :) + C.* rand(1, nvar).*( sr_R(i, :) - WATER_R(j, :));
%----------- Imaginary part update --------
      new_WATER_I= WATER_I(j, :) + C.* rand(1,nvar).*( sr_I(i, :) - WATER_I(j, :));
      sq_new_WATER = sqrt(new_WATER_R.^2 + new_WATER_I.^2);
      new_WATER =
sign(sin(new_WATER_I/sq_new_WATER))*sq_new_WATER+0.5*(UB+LB);
```

```
if objective_function(WATER(j, :),Fnum) > objective_function(new_WATER, Fnum)
        WATER(j, :) = new_WATER;
        WATER_R(j, :) = new_WATER_R;
        WATER_I(j, :) = new_WATER_I;
end
end
end
%------------Rivers flow to the sea --------------------------
for i = 1:Nsr_k
        new_sr = [];
%----------- Real component update --------
        new_sr_R = sr_R(i, :) + C.*rand(1,nvar).*(sea_R - sr_R(i,:));
%----------- Imaginary part update --------
        new_sr_I = sr_I(i, :) + C.*rand(1, nvar).*(sea_I - sr_I(i, :));
        sq_new_sr = sqrt(new_sr_R.^2 + new_sr_I.^2);
        new_sr = sign(sin(new_sr_I/sq_new_sr))*sq_new_sr + 0.5*(UB + LB);
    if objective_function(sr(i, :), Fnum) > objective_function(new_sr, Fnum)
        sr_R(i, :) = new_sr_R;
        sr_I(i, :) = new_sr_I;
        sr(i, :) = new_sr;
    end
end
%------------ Replacement of rivers and streams --------------------
    new_sr = [];
for i = 1:Nsr_k
        for j=(zigma_NCn(i)+1):zigma_NCn(i+1)
            if objective_function(sr(i,:), Fnum)>objective_function(WATER(j,:),Fnum)
                temp = WATER(j, :);
                temp_R = WATER_R(j, :);
                temp_I = WATER_I(j, :);
                WATER(j, :) = sr(i, :);
                WATER_R(j, :) = sr_R(i, :);
                WATER_I(j,:) = sr_I(i,:);
                sr(i, :) = temp;
                sr_R(i, :) = temp_R;
                sr_I(i, :) = temp_I;
                end
        end
end
%------------- River and sea location update ----------------------
for i = 1:Nsr_k
        if objective_function(sea, Fnum)>objective_function(sr(i, :), Fnum)
            temp = sr(i, :);
            temp_R = sr_R(i, :);
```

```
            temp_I = sr_I(i, :);
            sr(i, :) = sea;
            sr_R(i, :) = sea_R;
            sr_I(i, :) = sea_I;
            sea = temp;
            sea_R = temp_R;
            sea_I = temp_I;
        end
    end
        Xmin = sea;
        Fmin = objective_function(sea,Fnum);
        f(ii) = Fmin;
    end
    end
%----------------------End-----------------------------------------------
```