Knowledge-Based Systems 172 (2019) 42–63

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A hyper-heuristic for improving the initial population of whale optimization algorithm

Mohamed Abd Elaziz [a,c], Seyedali Mirjalili [b,*]

[a] Department of Mathematics, Faculty of Science, Zagazig University, Egypt
[b] Institute of Integrated and Intelligent Systems, Griffith University, Nathan, Brisbane, QLD 4111, Australia
[c] School of Computer Science & Technology, Huazhong university of Science and Technology, Wuhan 430074, China

ARTICLE INFO

Article history:
Received 15 August 2018
Received in revised form 16 November 2018
Accepted 10 February 2019
Available online 16 February 2019

Keywords:
Whale Optimization Algorithm (WOA)
Differential Evolution (DE)
Global optimization
Swarm intelligent

ABSTRACT

This paper improves the performance of the recently-proposed Whale Optimization Algorithm (WOA). WOA is a meta-heuristic that simulates the foraging behavior of humpback whales. There are several improvements in the literature for this algorithm of which chaotic maps and Opposition-Based Learning (OBL) are proved to be the most effective. In the former method, however, there are many chaotic maps that make it difficult to choose the best one for a given optimization algorithm. In the latter method, OBL should be applied to a portion of solutions in the population, which is normally obtained manually, which is time-consuming. This work proposed a hyper-heuristic to alleviate these drawbacks by automatically choosing a chaotic map and a portion of the population using the Differential Evolution (DE) algorithm. The proposed algorithm, which called DEWCO, has high ability to improve the exploration and local optima avoidance of WOA. In order to investigate the performance of the proposed DEWCO algorithm, several experiments are conducted on 35 standard CEC2005 functions and using seven algorithms. The experimental results show the superior performance of the proposed DEWCO algorithm to determine the optimal solutions of the test function problems.

1. Introduction

Global Optimization became the way for dealing with many problems in several fields such as parameter estimation [1], classification [2], scheduling [3], feature selection [4] and medical image [5,6], to name a few. Traditional optimization techniques can solve such problems, but they struggle in finding global optima when the problem becomes more challenging. Most of such techniques are exact and required gradient information of the problem. All these issues make them unreliable for solving challenging problems. On the other hand, Meta-Heuristics (MH) benefits from stochastic components and do not need gradient information. Therefore, they are better able to solve challenging problems. The MH techniques mostly simulate the physical or the biological behavior from nature [7,8]. Among these MH methods, those inspired from the intelligence of natural swarms mimic are very popular. Some of the examples are: social spider optimization (SSO) [9], grey wolf optimization (GWO) and [10], salp swarm algorithm (SSA) [11], Grasshopper optimization algorithm (GOA) [12], selfish optimization algorithm (SHO) [13]. Most of MH methods, the population-based, have high ability to avoid the trapped in a local optimum point since the solutions working together to jump out the stagnation point. In addition, the MH algorithms have common characteristics during the optimization process which aim to balance between exploration and exploitation. The main behavior of MH methods in the exploration is to find the more promising regions of the search domain. Meanwhile, exploitation evaluate the ability of the algorithm to determine the optimal solutions within the search space.

Recently, a new swarm-based algorithm called Whale Optimization Algorithm (WOA) [23] is proposed that mimics the social behavior of humpback whales. The WOA has several properties that give it a good ability to find the best solution for optimization problems, and it has shown its superior performance in several applications including bioinformatics [24], economic dispatch problem [25], workflow planning of construction sites [26], Liver segmentation in MRI images [27], image segmentation [28], and content-based image retrieval [29]. In addition, Sharawi et al. [30] introduced a feature selection system based on WOA. Also, Reddy in [31] proposed a novel technique to solve the feeder reconfiguration problem with an objective of minimizing total real loss and maximizing energy savings in a radial distribution system through using the whale optimization algorithm. Dinakara et al. in [32] used WOA to determine the optimal distributed generator (DG) resources' size. These resources were the small-scale electric power generating plants that can provide power to homes, businesses or

* Corresponding author.
E-mail addresses: meahmed@zu.edu.eg (M.A. Elaziz), seyedali.mirjalili@griffithuni.edu.au (S. Mirjalili).

https://doi.org/10.1016/j.knosys.2019.02.010
0950-7051/

Table 1
The description of chaotic maps.

NO.	Map	Formula	Condition
1	Chebyshev [14]	$x_{k+1} = \cos(k\cos^{-1}(x_k))$	–
2	Circle [15]	$x_{k+1} = x_k + b - (\frac{a}{2\pi})\sin(2\pi x_k) \bmod$	a = 0.5 and b = 0.2, it generates chaotic sequence in (0, 1).
	Gauss/Mouse [16]	$x_{k+1} = \begin{cases} 0 & , \quad x_k = 0 \\ \frac{1}{x_k \bmod (1)}, & otherwise \end{cases}$ $\frac{1}{x_k \bmod (1)} = \frac{1}{x_k} - \left[\frac{1}{x_k}\right]$	Generates chaotic sequences in (0, 1).
4	Iterative [17]	$x_{k+1} = \sin(\frac{a\pi}{x_k})$	$a \in (0, 1)$ is a suitable parameter.
5	Logistic [18]	$x_{k+1} = ax_k(1 - x_k)$	is the kth chaotic number, and $x_o \in (0, 1)$.
6	Piecewise [19]	$x_{k+1} = \begin{cases} \frac{x_k}{P}, & 0 \le x_k \prec P \\ \frac{x_k - P}{0.5 - P} & P \le x_k \prec 0.5 \\ \frac{1 - P - x_k}{0.5 - P}, & 0.5 \le x_k \prec 1 - P \\ \frac{1 - x_k}{P}, & 1 - P \le x_k \prec 1 \end{cases}$	$P \in (0, 0.5)$ is the control parameter and $x \in (0, 1)$ and $P \neq 0$.
7	Sine [20]	$x_{k+1} = \frac{a}{4}\sin(\pi x_k)$	$0 \prec a \prec 4$.
8	Singer [21]	$x_{k+1} = \mu(7.86x_k - 23.31x_k^2 + 28.75x_k^3 - 13.3x_k^4)$	–
9	Sinusoidal [18]	$x_{k+1} = ax_k^2\sin(\pi x_k)$	μ is a parameter between 0.9 and 1.08.
10	Tent [22]	$x_{k+1} = \begin{cases} \frac{x_k}{0.7}, & x_k \prec 0.7 \\ \frac{10}{3}(1 - x_k), & x_k \ge 0.7 \end{cases}$	–

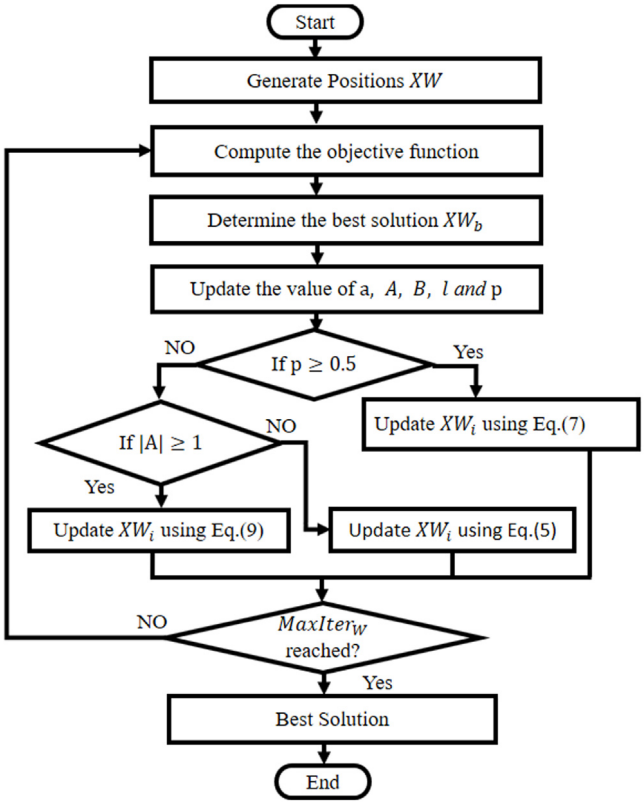


Fig. 1. The flowchart of the WOA.

industrial facilities in the distribution system. In addition to, Nasiri et al. [33] proposed a clustering method based on WOA and showed the merits of WO in the area of clustering too.

Despite the success of the original version of WOA, several works argued that its performance might degrade when solving some problems. For instance, in [34], Amolkumar et al. combined the WOA with the exponential grey wolf optimizer for data clustering. The Simulated Annealing was used in [35] as a local search method to improve the WOA to select the optimal subset of features to improve the accuracy of classification. In [36], Ling et al. used the Lévy Flight to improve the WOA for global optimization. Kaveh et al. presented an enhanced WOA for Sizing Optimization of Skeletal Structures. Also, the WOA algorithm with inertia weight was proposed in [37], in which the authors hybridized WOA with a pattern search technique to solve optimal power flow problems.

Most of the above-mentioned works incorporated local searches or methods to accelerate convergence that might lead the algorithm towards a local solution. Another popular method to improve the performance of algorithms is to use chaotic maps to provide systematic stochastic behaviors [38]. This gives a good ability to the WOA to search about the optimal solution with avoiding the problem of the stagnation to local point, and might even increase the convergence towards the global solution if tune properly.

In general, chaotic maps are applicable for use in computational applications such as optimization problems, especially for initialization the population. Many algorithms were enhanced by chaotic maps in the literature. For instance, Ewees et al. [39] provided a chaotic Multi-verse Optimization algorithm (CMVO) to improve the performance of the MVO algorithm. Assarzadeh et al. [40] proposed a chaotic PSO with mutation-based classifier PSO to classify patterns of different classes in the feature space. In [41], Shen proposed a new multi-swarm optimization algorithm with a chaotic mapping strategy based on PSO.

Chaotic maps have been used to improve WOA as well [42]. For example, in [43], the chaotic WOA was applied for transient stability constrained optimal power flow, where the chaotic maps were used to update the main stochastic operators of WOA. A similar idea was presented in [44] to solve in the area of photovoltaic cells.

In addition to chaotic maps strategy, another popular strategy called opposition-based learning (OBL) [45] has been used for enhancing the behaviors of the most swarm algorithms. The main intention of OBL is to generate the 'opposite' local of solutions in the initialization phase or during the optimization process to improve the exploration of the algorithm. In general, the OBL scheme computes the opposite solution for each solution in the population of solutions [46–48]. Based on the characteristics of the OBL strategy, Alamri et al. proposed a modified for WOA using OBL strategy [49]. Elaziz et al. [50] presented the OBL-based WOA strategy to generate the initial population and during the updated stage of the solution. This algorithm was used to find the optimal value of the parameter of solar cells diode models.

Despite the success of chaotic maps and OBL-based initialization, there are many chaotic maps and OBL methods which can be used to improve the WOA. This leads to a new optimization problem, in which an optimal set of chaotic maps OBL strategies should be found to maximize the performance of WOA. Due to a large number of possibilities, this problem is NP-hard and time-consuming problem. We identify this problem as a gap in the

Table 2
The definition of CEC2005 Benchmark.

ID	Formula of function	*LW*	*UW*	dim_W	Type
F1	$f(x) = \sum_{i=1}^{n} x_i^2$	−100	100	30	Unimodal
F2	$f(x) = \sum_{i=1}^{n} \|x_i\| + \Pi_{i=1}^{n} \|x_i\|$	−10	10	30	Unimodal
F3	$f(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_i)^2$	−100	100	30	Unimodal
F4	$f(x) = max_i\{\|x_i\|, 1 \leqslant i \leqslant n\}$	−100	100	30	Unimodal
F5	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	−30	30	30	Unimodal
F6	$f(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	−100	100	30	Unimodal
F7	$f(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1]$	−1.28	1.28	30	Unimodal
F8	$f(x) = \sum_{i=1}^{n} i x_i^2$	−10	10	30	Unimodal
F9	$f(x) = \sum_{i=1}^{n} i x_i^4$	−1.28	1.28	30	Unimodal
F10	$f(x) = \sum_{i=1}^{n} \|x_i\|^{i+1}$	−1	1	30	Unimodal
F11	$f(x) = \sum_{i=1}^{n} -x_i sin(\sqrt{\|x_i\|})$	−500	500	30	Multimodal
F12	$f(x) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	−5.12	5.12	30	Multimodal
F13	$f(x) = -20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	−32	32	30	Multimodal
F14	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \Pi_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	−600	600	30	Multimodal
F15	$f(x) = \frac{\pi}{n}\{10sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	−50	50	30	Multimodal
	$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leqslant x_i \leqslant a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$				
F16	$f(x) = 0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	−50	50	30	Multimodal
F17	$f(x) = \sum_{i=1}^{n}(x_i - 1)^2 + [1 + \sin^2(3\pi x_i + 1)] + \sin^2(3\pi x_1) + \|x_n - 1\|[1 + \sin^2(3\pi x_n)]$	−10	10	30	Multimodal
F18	$f(x) = \sum_{i=1}^{n} \|x_i sin(x_i) + 0.1x_i\|$	−10	10	30	Multimodal
F19	$f(x) = 0.1n - 0.1\sum_{i=1}^{n} \cos(5\pi x_i) - \sum_{i=1}^{n} x_i^2$	−1	1	30	Multimodal
F20	$f(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$	−5	10	30	Multimodal
F21	$f(x) = \sum_{i=1}^{n} 0.5 + \frac{\sin^2(\sqrt{100x_{i-1}^2 + x_i^2} - 0.5)}{1 + .001(x_i^2 - 2x_{i-1}x_i + x_i^2)^2}$	−5	10	30	Multimodal
F22	$f(x) = 0.1\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2 + \sin^2(3\pi x_{i+1}) + (x_n - 1)^2(1 + \sin^2(3\pi x_n))$	−5	5	30	Multimodal
F23	$f(x) = \sum_{i=1}^{n} (10^6)^{(i-1)/(n-1)} x_i^2$	−100	100	30	Multimodal
F24	$f(x) = 1 - cos(2\pi\sqrt{\sum_{i=1}^{n} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{n} x_i^2}$	−100	100	30	Multimodal
F25	$f(x) = 0.5 + \frac{\sin^2(\sqrt{\sum_{i=1}^{n} x_i^2}) - 0.5}{(1 + 0.001(\sum_{i=1}^{n} x_i^2))^2}$	−100	100	30	Multimodal
F26	$f(x) = (\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6})^{-}1$	−65.536	65.536	2	Multimodal
F27	$f(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	−5	5	4	Multimodal
F28	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - x_2^2 + 4x_2^4$	−5	5	2	Multimodal
F29	$f(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})cosx_1 + 10$	−5	5	2	Multimodal
F30	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$	−2	2	2	Multimodal
	$\times[30 + (2x_1 - 3x_2) \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$				
F31	$f(x) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	1	3	3	Multimodal
F32	$f(x) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	0	1	6	Multimodal
F33	$f(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	0	10	4	Multimodal
F34	$f(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	0	10	4	Multimodal
F35	$f(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	0	10	4	Multimodal

literature that motivated out attempts to proposed a systematic method for finding an optimal configuration of chaotic maps and OBL methods. We also target finding an optimal portion of the population to apply OBL. In general, the OBL is used in the initial stage to generate the initial population that has a large effect on the performance of the MH algorithm(s). So, the OBL is used to improve the initial population by computing the opposite direction of each solution and this coverage large area from the feasible domain. In addition, the process of randomly generating the set of solutions has a chance of producing or even reproducing the solutions from unproductive areas of the search domain. This chance is decreased when the OBL is used, also, it has been proved that the opposite

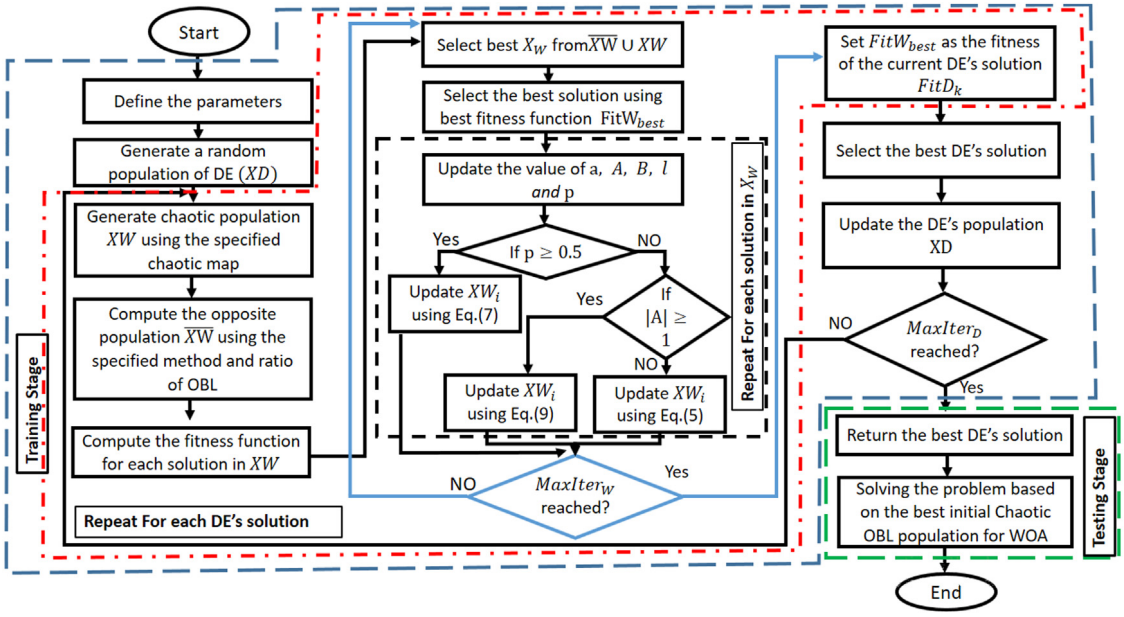


Fig. 2. The framework of the proposed algorithm.

Table 3
The average of the fitness value for each algorithm.

	WOA	DEWCO	GWO	SCA	ABC	SSO	MFO	MVO
F1	3.84E−11	**2.52E−17**	3.78E−2	5.87E+3	4.63E+2	3.82E+1	1.11E+4	3.98E+1
F2	2.38E−8	**8.05E−11**	4.54E−2	6.67E+0	7.67E−1	9.02E+0	6.60E+1	7.11E+6
F3	9.84E+4	**3.49E−2**	3.87E+2	3.65E+4	3.73E+4	1.37E+3	3.99E+4	7.85E+3
F4	5.74E+1	**6.48E−3**	2.13E+0	7.07E+1	7.04E+1	1.37E+1	7.30E+1	2.82E+1
F5	2.88E+1	**2.82E+1**	3.17E+1	1.79E+7	1.37E+3	1.70E+3	1.56E+7	1.02E+4
F6	2.22E+0	**3.22E−1**	3.58E+0	4.20E+3	1.03E+3	1.44E+2	1.13E+4	4.99E+1
F7	3.55E−2	**1.09E−3**	1.59E−2	8.05E−2	2.29E−1	4.48E−2	2.24E−1	7.41E−2
F8	1.57E−10	**1.01E−17**	4.45E−3	5.11E+2	8.92E+1	1.21E+1	1.70E+3	3.44E+1
F9	7.01E−20	**3.59E−28**	1.57E−8	4.44E+0	5.54E−3	3.43E−1	1.10E+1	5.61E−4
F10	5.51E−14	**1.16E−16**	1.93E−13	1.23E−1	9.20E−3	9.21E−3	1.78E−2	2.96E−3
F11	−1.63E+3	**−1.63E+3**	−7.58E+2	−7.73E+2	−1.13E+3	−4.65E+2	−1.32E+3	−9.37E+2
F12	3.89E+0	**3.87E−14**	3.34E+1	1.46E+2	9.72E+1	1.07E+2	2.24E+2	1.70E+2
F13	9.52E−7	**2.71E−9**	4.76E−2	1.89E+1	1.41E+1	3.48E+0	1.92E+1	3.72E+0
F14	3.68E−2	**7.55E−17**	1.31E−1	3.67E+1	9.65E+0	2.70E+0	1.12E+2	1.43E+0
F15	2.10E−1	**1.23E−2**	5.01E−1	4.70E+7	1.16E−1	2.09E+1	2.01E+7	1.33E+1
F16	1.37E+0	**1.25E−1**	2.32E+0	8.11E+7	2.39E−1	1.89E+1	6.77E+7	2.53E+1
F17	7.99E+0	**1.20E+0**	1.05E+1	8.59E+1	1.83E+1	2.21E+1	2.31E+2	3.71E+1
F18	2.72E+0	**9.49E−13**	4.41E−1	1.18E+1	6.60E+0	5.71E+0	1.81E+1	1.67E+1
F19	3.14E−14	**4.09E−16**	5.25E−5	1.33E+0	1.49E+0	2.89E+0	3.12E+0	1.73E+0
F20	5.04E+2	**1.07E−1**	3.52E+1	1.93E+2	3.38E+2	1.44E+2	5.03E+2	1.42E+2
F21	−2.64E+1	−2.07E+1	**3.70E+0**	5.55E+0	−5.69E+0	4.39E+0	−6.42E+0	7.32E+0
F22	5.10E+0	1.05E+0	7.48E+0	3.83E+1	5.15E+0	3.71E+0	5.81E+1	**4.76E−1**
F23	2.51E−6	**2.91E−15**	7.47E+1	2.99E+6	2.18E+5	1.17E+6	4.84E+7	3.18E+7
F24	1.44E−1	**1.20E−2**	7.36E−1	6.74E+0	1.44E+1	3.80E+0	1.35E+1	2.56E+0
F25	4.36E−2	**3.89E−4**	2.09E−1	4.93E−1	5.00E−1	4.42E−1	5.00E−1	4.38E−1
F26	5.4139	5.4166	5.9754	4.2002	**0.9980**	**0.9980**	2.4225	6.2455
F27	0.0020	**0.0003**	0.0054	0.0013	0.0015	0.0007	0.0022	0.0057
F28	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0314**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**
F29	0.3999	**0.3979**	**0.3979**	0.4044	**0.3979**	**0.3979**	**0.3979**	**0.3979**
F30	7.5326	**3.0001**	6.2413	3.0035	3.0198	**3.0006**	4.0800	6.2401
F31	−3.8272	**−3.8627**	**−3.8627**	−3.8583	−3.8628	−3.8621	−3.8623	**−3.8627**
F32	−3.1409	−3.2883	−3.3216	−3.0711	**−3.3220**	−3.2940	−3.3191	−3.2817
F33	−6.1688	−7.6230	−8.3102	−1.9438	−9.9143	**−10.1125**	−7.0471	−6.3320
F34	−6.0539	−7.8193	−9.2688	−2.2698	−10.0849	**−10.3616**	−7.8615	−6.7151
F35	−5.6548	−7.3771	−9.8404	−2.3465	−10.3474	**−10.5322**	−7.5658	−7.4573

numbers have a high probability to be closer to the optimal solution than those random solutions [51].

In order to achieve this goal, a Hyper-heuristic (HH) is developed [52,53]. In general, HH methods consists of two levels: Low-Level and High-Level. The low-level heuristic (LLH) class contains a set of Low-level heuristics (i.e., local search methods), which benefit from each method, as well as, they are easy to implemented and fast run time. The high-level hyper-heuristics are considered as the controller of LLH methods [54]. Due to the better ability to converge to the (nearly) optimal solutions and problem independence [55–57], such hyper-heuristics have been applied to many real-world applications: software maintenance [58], space allocation and timetabling problems [59], traveling salesman problems [60], dynamic optimization problems [61], determining shipper sizes for storage and transportation [62], and scheduling problem [63].

Table 4
The best fitness value for each algorithm.

	WOA	DEWCO	GWO	SCA	ABC	SSO	MFO	MVO
F1	5.740E−14	**1.906E−26**	1.659E−2	8.191E+1	1.927E+0	3.816E+1	5.505E+3	2.186E+1
F2	5.562E−11	**7.989E−15**	1.879E−2	7.814E−1	2.586E−1	9.024E+0	4.719E+1	1.633E+1
F3	6.372E+4	**1.398E−6**	1.295E+2	1.091E+4	2.240E+4	1.366E+3	2.708E+4	4.376E+3
F4	7.221E+0	**1.239E−6**	1.046E+0	5.649E+1	5.788E+1	1.372E+1	5.965E+1	9.977E+0
F5	2.871E+1	**2.772E+1**	2.938E+1	4.685E+5	2.337E+2	1.703E+3	4.976E+6	9.739E+2
F6	8.503E−1	**1.242E−1**	2.123E+0	6.414E+2	7.019E+0	1.442E+2	5.566E+3	2.726E+1
F7	1.465E−3	**5.761E−5**	4.761E−3	2.161E−2	1.435E−1	4.485E−2	1.409E−1	2.303E−2
F8	9.819E−15	**2.910E−23**	8.681E−4	3.499E+1	1.901E−1	1.208E+1	9.111E+2	8.137E+0
F9	1.456E−27	**1.104E−40**	9.692E−11	6.285E−2	3.961E−7	3.430E−1	2.460E+0	6.365E−5
F10	1.343E−19	**2.176E−25**	2.662E−16	3.411E−3	1.990E−4	9.206E−3	4.797E−3	2.938E−4
F11	−1.632E+3	**−1.632E+3**	−9.017E+2	−8.871E+2	−1.236E+3	−4.645E+2	−1.455E+3	−1.204E+3
F12	**0.000E+0**	**0.000E+0**	1.696E+1	6.094E+1	6.612E+1	1.071E+2	1.765E+2	9.037E+1
F13	4.824E−10	**3.323E−12**	2.030E−2	1.178E+1	1.097E+1	3.478E+0	1.521E+1	2.709E+0
F14	1.188E−14	**0.000E+0**	7.009E−3	1.033E+1	1.046E+0	2.704E+0	2.494E+1	1.276E+0
F15	1.679E−2	2.722E−3	1.535E−1	7.770E+4	**1.157E−3**	2.085E+1	1.762E+6	4.995E+0
F16	8.112E−1	4.103E−2	1.477E+0	1.742E+6	**4.027E−2**	1.886E+1	1.071E+7	2.062E+0
F17	1.644E+0	1.713E−1	2.327E+0	2.493E+1	**9.700E−2**	2.208E+1	1.007E+2	1.427E+0
F18	1.499E−10	**2.971E−16**	1.577E−2	6.383E−1	4.335E+0	5.714E+0	1.004E+1	1.130E+1
F19	**0.000E+0**	**0.000E+0**	1.940E−5	5.724E−3	6.801E−1	2.886E+0	2.196E+0	8.243E−1
F20	3.162E+2	**1.162E−5**	1.297E+1	5.096E+1	2.242E+2	1.444E+2	2.381E+2	6.759E+1
F21	−2.900E+1	−2.601E+1	−3.612E+0	3.528E+0	−1.115E+1	4.388E+0	−1.129E+1	**3.426E+0**
F22	2.085E+0	**8.615E−2**	2.094E+0	2.431E+1	6.850E−2	3.707E+0	2.724E+1	1.097E−1
F23	2.953E−11	**6.210E−23**	1.304E+1	2.198E+5	4.250E+3	1.165E+6	9.351E+6	6.083E+6
F24	1.133E−7	**2.584E−13**	3.999E−1	1.933E+0	1.042E+1	3.800E+0	9.710E+0	2.100E+0
F25	9.284E−10	**0.000E+0**	7.819E−2	4.642E−1	4.996E−1	4.419E−1	4.995E−1	3.455E−1
F26	9.980E−1	9.980E−1	9.980E−1	**1.003E+0**	9.980E−1	9.980E−1	9.980E−1	9.980E−1
F27	3.241E−4	**3.085E−4**	3.821E−4	6.311E−4	6.695E−4	7.308E−4	7.352E−4	6.344E−4
F28	**−1.032E+0**	**−1.032E+0**	**−1.032E+0**	**−1.032E+0**	**−1.032E+0**	**−1.032E+0**	**−1.032E+0**	**−1.032E+0**
F29	**3.979E−1**	**3.979E−1**	**3.979E−1**	3.983E−1	**3.979E−1**	**3.979E−1**	**3.979E−1**	**3.979E−1**
F30	**3.000E+0**	**3.000E+0**	**3.000E+0**	**3.000E+0**	**3.000E+0**	**3.001E+0**	**3.000E+0**	**3.000E+0**
F31	**−3.863E+0**	**−3.863E+0**	−3.863E+0	**−3.862E+0**	**−3.863E+0**	**−3.862E+0**	**−3.863E+0**	**−3.863E+0**
F32	−3.315E+0	**−3.321E+0**	**−3.322E+0**	−3.247E+0	**−3.322E+0**	−3.294E+0	**−3.322E+0**	**−3.322E+0**
F33	−1.014E+1	−1.011E+1	−1.014E+1	−4.462E+0	−1.015E+1	−1.011E+1	**−1.015E+1**	−1.015E+1
F34	−1.038E+1	−1.034E+1	−1.040E+1	−5.197E+0	−1.040E+1	−1.036E+1	**−1.040E+1**	−1.040E+1
F35	−1.043E+1	−1.053E+1	−1.053E+1	−4.410E+0	−1.053E+1	−1.053E+1	**−1.054E+1**	−1.053E+1

Table 5
The worst fitness value for each algorithm.

	WOA	DEWCO	GWO	SCA	ABC	SSO	MFO	MVO
F1	2.763E−10	**6.049E−16**	1.081E−1	1.626E+4	2.794E+3	3.816E+1	2.023E+4	5.806E+1
F2	2.081E−7	**1.296E−9**	7.516E−2	1.874E+1	1.577E+0	9.024E+0	1.074E+2	1.774E+8
F3	1.510E+5	**4.083E−1**	1.268E+3	7.257E+4	5.115E+4	1.366E+3	6.533E+4	1.302E+4
F4	8.802E+1	**3.416E−2**	4.320E+0	8.512E+1	7.641E+1	1.372E+1	8.590E+1	5.878E+1
F5	2.888E+1	**2.878E+1**	3.713E+1	5.537E+7	1.045E+4	1.703E+3	8.316E+7	4.889E+4
F6	3.320E+0	**7.026E−1**	5.190E+0	9.785E+3	4.515E+3	1.442E+2	2.128E+4	8.756E+1
F7	1.543E−1	**4.511E−3**	3.628E−2	1.592E−1	3.350E−1	4.485E−2	3.535E−1	1.148E−1
F8	1.338E−9	**9.085E−17**	1.009E−2	1.275E+3	7.734E+2	1.208E+1	2.656E+3	8.351E+1
F9	1.120E−18	**5.341E−27**	2.899E−7	1.344E+1	1.295E−1	3.430E−1	3.024E+1	1.546E−3
F10	1.126E−12	**2.860E−15**	3.533E−12	5.213E−1	3.830E−2	9.206E−3	5.248E−2	7.253E−3
F11	**−1.632E+3**	**−1.632E+3**	−6.467E+2	−6.415E+2	−1.037E+3	−4.645E+2	−1.106E+3	−7.882E+2
F12	9.547E+1	**1.705E−13**	6.163E+1	2.350E+2	1.149E+2	1.071E+2	3.012E+2	2.302E+2
F13	5.588E−6	**5.746E−8**	1.056E−1	2.046E+1	1.628E+1	3.478E+0	2.005E+1	5.146E+0
F14	9.199E−1	**7.772E−16**	2.860E−1	1.064E+2	5.042E+1	2.704E+0	2.665E+2	1.663E+0
F15	7.758E−1	**2.550E−2**	1.064E+0	3.727E+8	5.333E−1	2.085E+1	9.603E+7	2.957E+1
F16	2.274E+0	**2.365E−1**	3.403E+0	2.714E+8	2.388E+0	1.886E+1	4.317E+8	7.639E+1
F17	1.398E+1	**2.110E+0**	1.836E+1	1.550E+2	6.727E+1	2.208E+1	4.128E+2	7.608E+2
F18	4.020E+1	**9.330E−12**	2.164E+0	3.191E+1	9.335E+0	5.714E+0	2.794E+1	3.018E+1
F19	2.340E−13	**1.776E−15**	1.226E−4	4.589E+0	2.214E+0	2.886E+0	3.952E+0	2.566E+0
F20	8.973E+2	**1.660E+0**	8.059E+1	3.579E+2	4.021E+2	1.444E+2	6.966E+2	2.061E+2
F21	−1.372E+1	−1.627E+1	7.166E+0	7.611E+0	**2.991E−1**	4.388E+0	−2.138E+0	9.076E+0
F22	1.301E+1	1.812E+0	1.272E+1	5.377E+1	1.750E+1	3.707E+0	1.117E+2	**1.436E+0**
F23	2.010E−5	**4.021E−14**	1.690E+2	6.240E+6	2.203E+6	1.165E+6	2.453E+8	6.647E+7
F25	3.999E−1	**9.987E−2**	8.999E−1	1.227E+1	1.716E+1	3.800E+0	1.970E+1	3.100E+0
F26	1.270E−1	**9.716E−3**	2.727E−1	4.995E−1	4.999E−1	4.419E−1	4.999E−1	4.903E−1
F27	1.550E+1	1.267E+1	1.830E+1	1.076E+1	**9.980E−1**	**9.980E−1**	7.874E+0	1.456E+1
F28	2.096E−2	**3.757E−4**	2.121E−2	2.267E−3	2.992E−3	7.308E−4	2.038E−2	2.081E−2
F29	**−1.032E+0**	**−1.032E+0**	**−1.032E+0**	−1.030E+0	**−1.032E+0**	**−1.032E+0**	**−1.032E+0**	**−1.032E+0**
F30	4.064E−1	3.982E−1	**3.980E−1**	4.169E−1	3.979E−1	3.979E−1	3.979E−1	3.979E−1
F31	3.388E+1	**3.001E+0**	8.400E+1	3.022E+0	3.224E+0	3.001E+0	3.000E+1	8.400E+1
F32	−3.090E+0	**−3.863E+0**	**−3.862E+0**	−3.854E+0	**−3.863E+0**	**−3.862E+0**	−3.857E+0	−3.862E+0
F33	−2.184E+0	−3.235E+0	**−3.321E+0**	−2.243E+0	−3.322E+0	−3.294E+0	−3.250E+0	−2.316E+0
F34	−2.056E+0	−5.023E+0	−2.665E+0	−4.971E−1	−8.523E+0	**−1.011E+1**	−2.630E+0	−2.629E+0
F35	−1.817E+0	−5.079E+0	−1.837E+0	−5.206E−1	−7.625E+0	**−1.036E+1**	−2.752E+0	−2.751E+0
F36	−1.672E+0	−5.110E+0	−2.420E+0	−9.331E−1	−9.009E+0	**−1.053E+1**	−2.422E+0	−2.421E+0

Table 6
The standard deviation of the fitness value for each algorithm.

	WOA	DEWCO	GWO	SCA	ABC	SSO	MFO	MVO
F1	6.54E−11	**1.21E−16**	2.16E−2	4.35E+3	7.43E+2	7.25E−15	3.92E+3	8.74E+0
F2	4.27E−8	2.64E−10	1.53E−2	3.92E+0	3.72E−1	**1.81E−15**	1.30E+1	3.55E+7
F3	2.67E+4	9.60E−2	2.63E+2	1.67E+4	6.75E+3	**6.96E−13**	9.97E+3	2.37E+3
F4	2.82E+1	9.01E−3	8.99E−1	7.25E+0	4.39E+0	**0.00E+0**	8.57E+0	1.02E+1
F5	3.90E−2	3.46E−1	1.94E+0	1.48E+7	2.14E+3	**2.32E−13**	1.78E+7	1.23E+4
F6	6.43E−1	1.25E−1	8.46E−1	2.62E+3	1.22E+3	**2.90E−14**	4.27E+3	1.46E+1
F7	3.66E−2	1.07E−3	7.74E−3	3.58E−2	4.97E−2	**1.42E−17**	6.53E−2	2.70E−2
F8	3.89E−10	2.15E−17	2.57E−3	3.67E+2	1.71E+2	**0.00E+0**	5.13E+2	2.03E+1
F9	2.43E−19	**1.25E−27**	5.73E−8	4.07E+0	2.58E−2	1.13E−16	8.01E+0	4.39E−4
F10	2.25E−13	5.72E−16	7.20E−13	1.31E−1	8.42E−3	**3.54E−18**	1.32E−2	1.70E−3
F11	**0.00E+0**	**0.00E+0**	7.88E+1	6.53E+1	5.03E+1	1.74E−13	1.01E+2	9.00E+1
F12	1.91E+1	6.30E−14	1.13E+1	4.19E+1	1.28E+1	**0.00E+0**	3.24E+1	3.32E+1
F13	1.44E−6	1.14E−8	2.39E−2	2.75E+0	1.38E+0	**4.53E−16**	1.13E+0	5.32E−1
F14	1.84E−1	**1.72E−16**	8.05E−2	2.43E+1	1.17E+1	1.81E−15	5.52E+1	1.09E−1
F15	1.56E−1	5.94E−3	2.43E−1	7.18E+7	1.36E−1	**7.25E−15**	1.89E+7	5.97E+0
F16	3.82E−1	4.89E−2	5.03E−1	7.09E+7	4.56E−1	**0.00E+0**	8.23E+7	1.83E+1
F17	3.02E+0	4.26E−1	3.61E+0	4.15E+1	2.02E+1	**0.00E+0**	8.64E+1	1.51E+2
F18	9.57E+0	2.42E−12	4.97E−1	6.50E+0	1.53E+0	**9.06E−16**	5.09E+0	3.92E+0
F19	6.74E−14	**7.13E−16**	3.11E−5	1.00E+0	4.23E−1	1.36E−15	4.62E−1	4.47E−1
F20	1.40E+2	3.69E−1	1.68E+1	6.30E+1	5.65E+1	**5.80E−14**	1.10E+2	4.24E+1
F21	3.90E+0	2.21E+0	2.37E+0	1.00E+0	3.47E+0	**2.72E−15**	1.97E+0	1.22E+0
F22	2.53E+0	4.21E−1	2.48E+0	7.57E+0	5.46E+0	**9.06E−16**	2.31E+1	3.98E−1
F23	5.60E−6	**8.75E−15**	4.14E+1	1.79E+6	4.64E+5	7.13E−10	4.79E+7	1.43E+7
F24	1.19E−1	3.31E−2	1.19E−1	2.41E+0	1.76E+0	**1.81E−15**	2.34E+0	2.73E−1
F25	3.41E−2	1.94E−3	4.19E−2	8.10E−3	6.55E−5	**0.00E+0**	1.19E−4	4.11E−2
F26	4.84E+0	4.79E+0	4.79E+0	3.44E+0	1.52E−6	**2.27E−16**	1.71E+0	4.21E+0
F27	4.08E−3	1.75E−5	8.64E−3	4.98E−4	6.12E−4	**0.00E+0**	4.06E−3	7.97E−3
F28	1.61E−6	3.27E−7	8.72E−7	2.76E−4	2.12E−7	**4.53E−16**	5.84E−16	1.41E−5
F29	2.66E−3	9.03E−5	3.83E−5	5.26E−3	1.57E−7	**0.00E+0**	0.00E+0	1.01E−5
F30	1.06E+1	2.31E−4	1.62E+1	5.70E−3	4.38E−2	**1.36E−15**	5.40E+0	1.62E+1
F31	1.54E−1	5.31E−5	1.27E−4	2.20E−3	2.74E−5	**9.06E−16**	1.49E−3	7.89E−5
F32	2.88E−1	2.05E−2	2.84E−4	2.54E−1	2.81E−6	**9.06E−16**	1.44E−2	2.01E−1
F33	2.55E+0	2.37E+0	3.01E+0	1.43E+0	3.80E−1	**3.63E−15**	3.41E+0	3.31E+0
F34	2.47E+0	2.49E+0	2.57E+0	1.38E+0	5.62E−1	**1.81E−15**	3.49E+0	3.43E+0
F35	2.87E+0	2.43E+0	2.23E+0	1.08E+0	2.91E−1	**3.63E−15**	3.75E+0	3.86E+0

Table 7
The CPU time (s) required by each algorithm.

	WOA	DEWCO	GWO	SCA	ABC	SSO	MFO	MVO	Time-train	After-train
F1	**0.048**	0.299	0.110	0.495	5.888	0.539	0.170	0.149	0.260	0.039
F2	**0.035**	0.267	0.047	0.166	0.319	0.210	0.046	0.064	0.236	0.030
F3	**0.088**	0.499	0.114	0.218	0.332	0.256	0.094	0.141	0.420	0.078
F4	**0.030**	0.255	0.104	0.064	0.437	0.183	0.039	0.049	0.231	0.024
F5	0.082	0.319	0.064	0.099	0.227	0.187	4.800	**0.054**	0.250	0.069
F6	**0.032**	0.252	0.049	0.046	0.288	0.184	0.037	0.051	0.225	0.027
F7	**0.042**	0.353	0.075	0.057	0.242	0.221	0.050	0.075	0.275	0.078
F8	0.050	0.344	0.044	0.041	0.144	0.179	**0.038**	0.055	0.272	0.072
F9	**0.047**	0.355	0.056	0.049	0.165	0.188	0.049	0.064	0.276	0.079
F10	0.063	0.374	0.061	0.053	0.182	0.196	**0.056**	0.068	0.292	0.082
F11	0.070	0.331	0.048	0.048	0.257	0.185	**0.040**	0.045	0.260	0.071
F12	**0.032**	0.312	0.044	0.052	0.196	0.227	0.040	0.054	0.247	0.065
F13	**0.033**	0.311	0.073	0.044	0.179	0.187	0.043	0.057	0.243	0.068
F14	**0.037**	0.328	0.052	0.053	0.173	0.186	0.045	0.073	0.256	0.071
F15	**0.074**	0.541	0.087	0.087	0.260	0.224	0.078	0.090	0.435	0.106
F16	**0.076**	0.483	0.088	0.080	0.300	0.228	0.121	0.096	0.373	0.110
F17	0.049	0.328	0.049	0.041	0.154	0.184	**0.046**	0.055	0.255	0.073
F18	0.040	0.299	0.046	0.040	0.150	0.189	**0.040**	0.055	0.235	0.064
F19	**0.034**	0.308	0.089	0.039	0.160	0.188	0.042	0.055	0.243	0.066
F20	**0.037**	0.329	0.050	0.046	0.154	0.186	0.044	0.054	0.257	0.072
F21	0.059	0.370	0.063	0.051	0.177	0.204	**0.051**	0.106	0.285	0.085
F22	**0.039**	0.325	0.049	0.051	0.163	0.182	0.049	0.054	0.254	0.071
F23	**0.051**	0.380	0.065	0.059	0.197	0.198	0.057	0.068	0.298	0.082
F24	0.041	0.299	0.059	0.047	0.155	0.180	**0.040**	0.051	0.233	0.066
F25	**0.032**	0.297	0.049	0.040	0.148	0.185	0.040	0.051	0.233	0.065
F26	0.159	0.744	0.174	**0.147**	0.411	0.370	0.154	**0.147**	0.576	0.168
F27	**0.029**	0.260	0.043	0.036	0.171	0.191	0.039	0.049	0.200	0.060
F28	0.036	0.218	0.030	0.066	0.186	0.181	0.063	0.032	0.190	0.028
F29	0.029	0.204	0.045	**0.025**	0.157	0.179	0.029	0.043	0.185	0.019
F30	0.029	0.207	0.041	0.026	0.142	0.184	**0.024**	0.032	0.188	0.019
F31	0.046	0.257	0.058	0.036	0.189	0.199	**0.034**	0.059	0.214	0.043
F32	0.070	0.277	0.087	**0.035**	0.277	0.223	0.036	0.042	0.215	0.062
F33	0.063	0.325	0.082	0.054	0.225	0.217	**0.053**	0.062	0.266	0.059
F34	0.077	0.365	0.064	0.063	0.239	0.270	**0.063**	0.074	0.295	0.071
F35	0.111	0.433	0.082	0.074	0.279	0.321	**0.077**	0.079	0.335	0.098

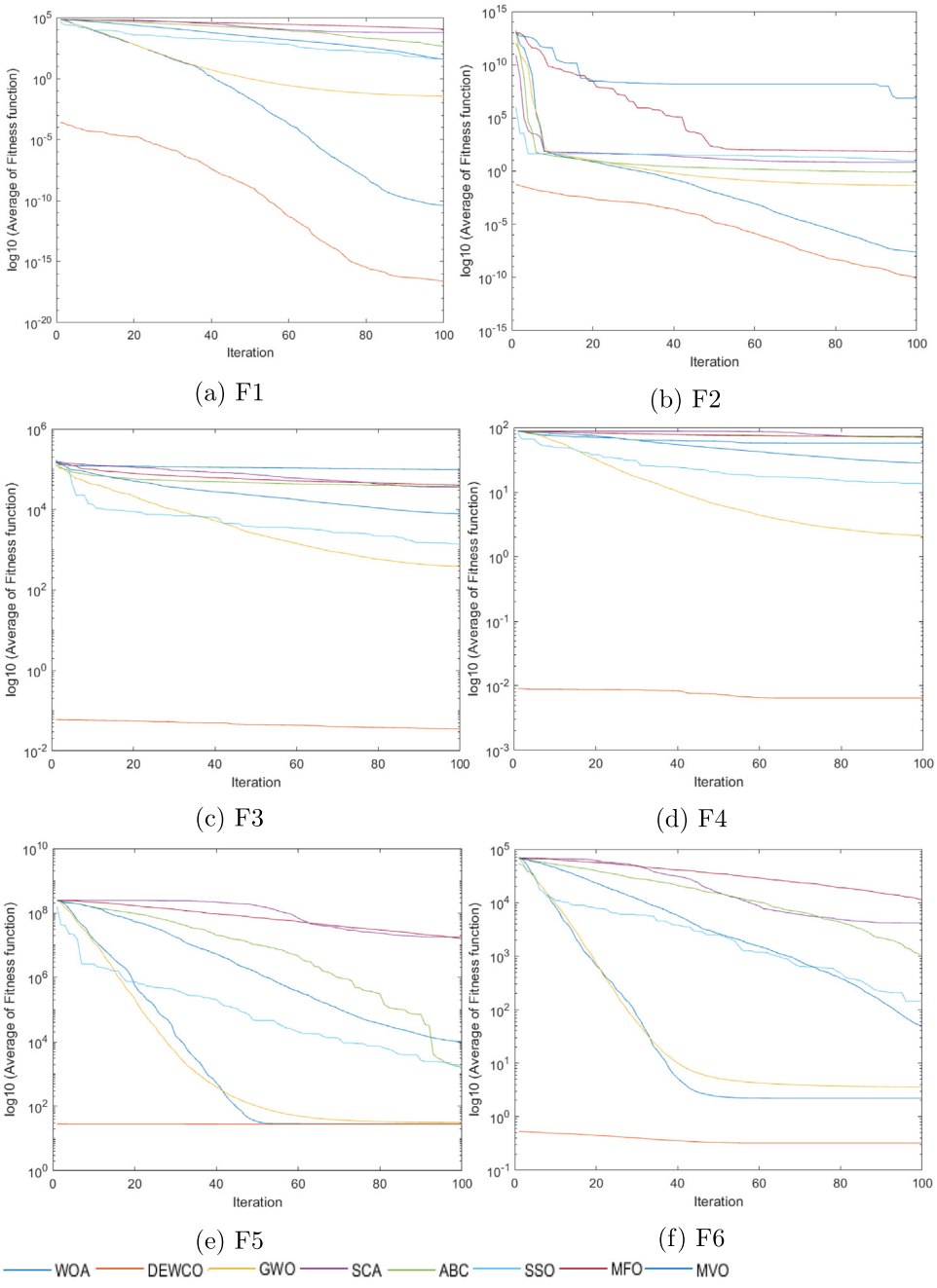


Fig. 3. The convergence curves of the algorithm along the functions F1–F6.

Based on the previously mentioned properties of the high-level methods, the hyper-heuristic is proposed using the DE algorithm [64] in this work to find an optimal configuration for WA. The DE is simple, yet effective meta-heuristic [64] with a few number of control parameters. In addition, according to the previous comprehensive studies [65–67], the DE algorithm has shown its superior performance among other meta-heuristics in terms of robustness and convergence when applied to global optimization problems and real-world problems. The main reason why we have chosen this algorithm is the fact that it has a large number of evolutionary operators that can borrowed to design a memetic algorithm. We have done several experiments and found that the operators used in this work are the most efficient ones. For example, Optimal power flow solutions [68], Economic and Emission Dispatch [69], and prediction of continuous blood glucose [70]. In this work, we investigate its performance as a hyper-heuristic too.

The proposed hyper-heuristic, which called DEWCO, consists of two stages. For once, an optimal configuration from the chaotic map, OBL, and the ratio of the population is selected using the DE algorithm. For another, this optimal configuration is used in the second stage to improve the performance of WOA to find the optimal solution of the given problem.

The mains contributions of this paper can summarized as the following:

- Proposing a hyper-heuristic using DE algorithm to find the optimal configuration from the chaotic maps, OBL method, and the ratio of population to find the optimal initial population to improve the WOA.
- Evaluate the performance of the proposed DEWCO algorithm using thirty-five functions from CEC2005 benchmark functions.

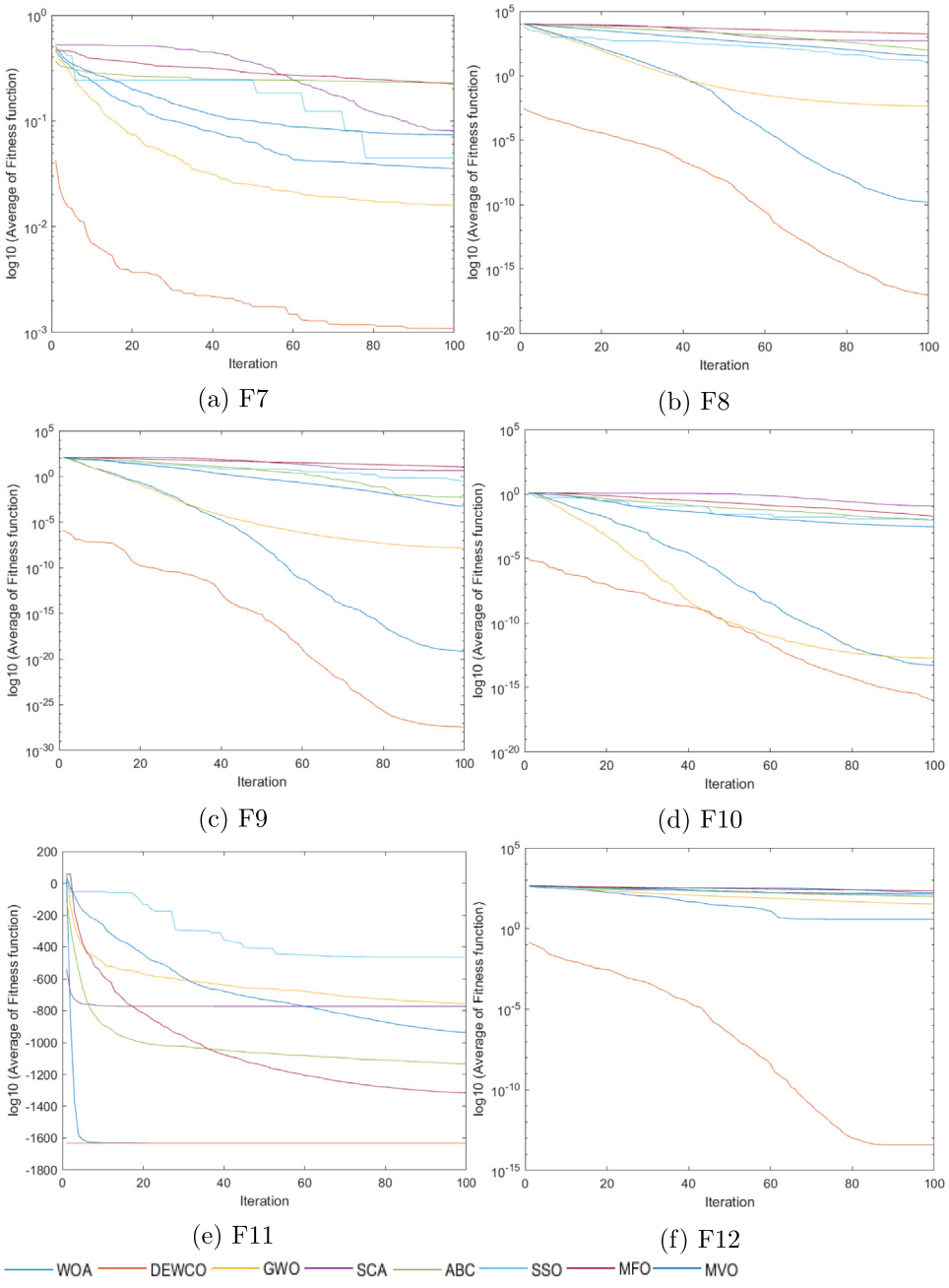


Fig. 4. The convergence curves of the algorithm along the functions F7–F12.

- The proposed DEWCO algorithm is compared with a set of state-of-the-art algorithms including Whale optimization algorithm (WOA) [71], Grey wolf optimizer (GWO) [10], Sine Cosine Algorithm (SCA) [72], Artificial Bee Colony (ABC) [73], Social-spider Optimization (SSO) [9], Moth Flame Optimization (MFO) [74], and Multi-Verse optimizer (MVO) [75].
- The influence of the population size, dimension, and the maximum number of iterations on the performance of the DEWCO algorithm.

The structure of the rest of this paper is organized as follows: The background of the Differential Evolution (DE), the chaotic maps, the opposition-based Learning (OBL) strategy, and the whale optimization (WOA) algorithm are discussed in Section 2. The proposed DEWCO algorithm is explained in Section 3 and a set of experimental series are provided in Section 4. The conclusion and the future works are given in Section 5.

2. Background

2.1. Differential evolution

The DE algorithm is considered as one of the most popular Evolutionary Computation (EC) algorithms proposed in [64]. It has three main operators, similar to other EC methods, called mutation, crossover, and selection.

In general, the DE starts by randomly generating a population X which contains N_D solutions for the test problem as given in the following equation:

$$XD_{ij} = L_j + rand_j \times (UD_j - LD_j), j = 1, 2, \ldots, dim_D,\ i = 1, 2, \ldots, N_D, \quad (1)$$

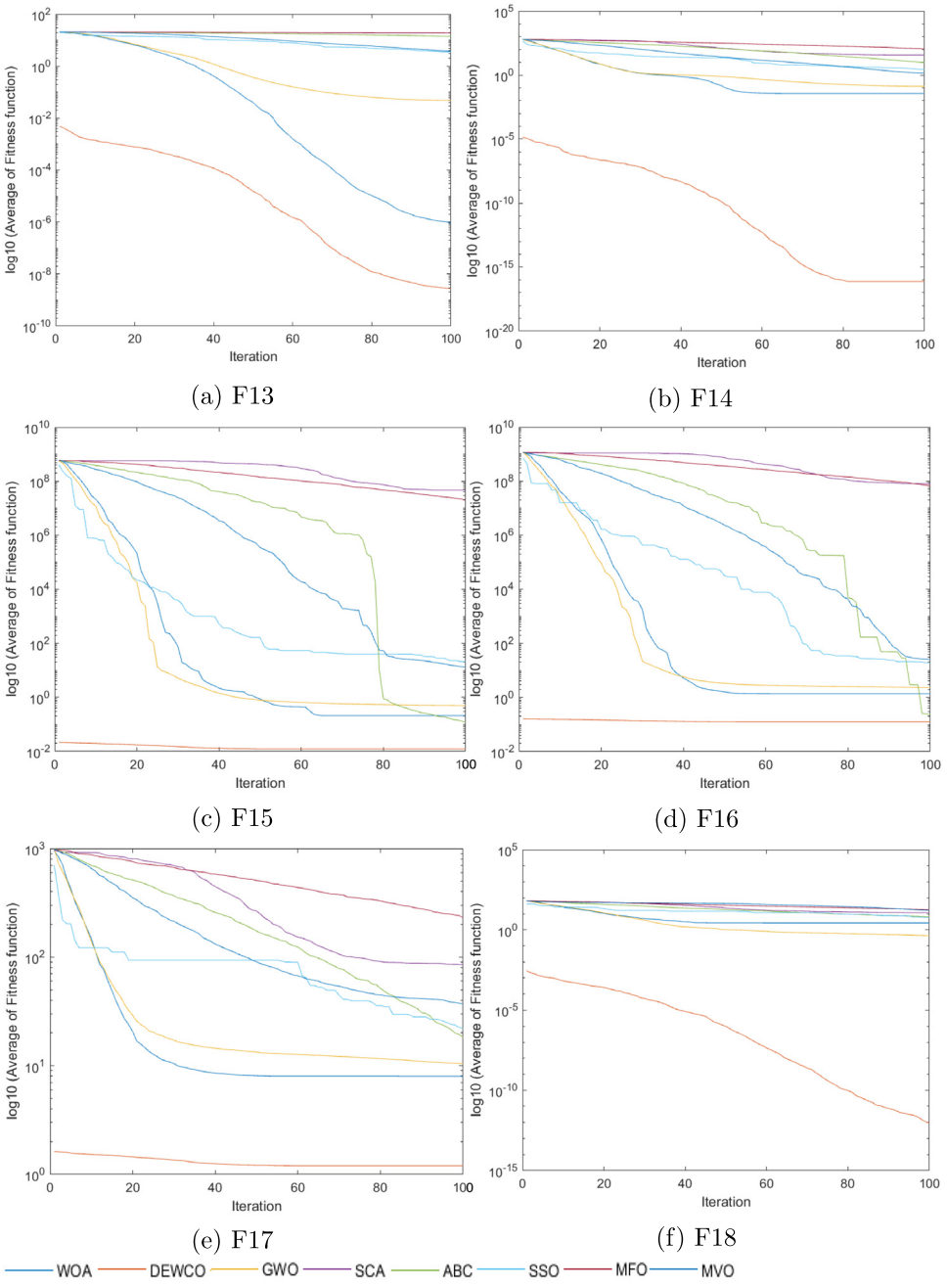


Fig. 5. The convergence curves of the algorithm along the functions F13–F18.

where $rand_j \in [0, 1]$ is a random number, LD_j, and UD_j are the lower and the upper boundary of the jth dimension of the XD_i. dim_D represents the dimension of the ith solution.

Each solution is then evaluated by computing its objective function, and the best solution is determined. The next step in the DE is to construct a new solution Z_i by using the mutation operation that defined as

$$Z_i(t) = XD_{r1}(t) + F * (XD_{r2}(t) - XD_{r3}(t)), \tag{2}$$

where mutation scaling parameter F, and $XD_{rk}, k = 1, 2, 3$ are the three solutions which chosen randomly from $[1, N_D]$. In general, the mutation operation represented in Eq. (2) is called the $DE/rand/1$ strategy. Thereafter, the crossover operation is performed to generate an offspring W_i from the current parent solution XD_i and Z_i as:

$$W_{ij}(t) = \begin{cases} Z_{ij}(t) & if\ \alpha_j \leq CR\ or\ \delta_i \\ XD_{ij}(t) & otherwise, \end{cases} \tag{3}$$

where CR represents the crossover probability. The α_j is a random value chosen for the jth dimension of XD_i, also δ_i represents a random index belong to $[1, dim_D]$.

The third operation is called selection operation, in which if the fitness function $Fit(W_i)$ of the offspring W_i is better than the fitness function $Fit(XD_i)$ of x_i, then it replaced the current parent XD_i, otherwise persevere XD_i to used the next generation. This operation can be represented as in the following equation:

$$XD_i(t+1) = \begin{cases} W_i(t) & if\ f(W_i(t)) \leq Fit(XD_i(t)) \\ XD_i(t) & otherwise, \end{cases} \tag{4}$$

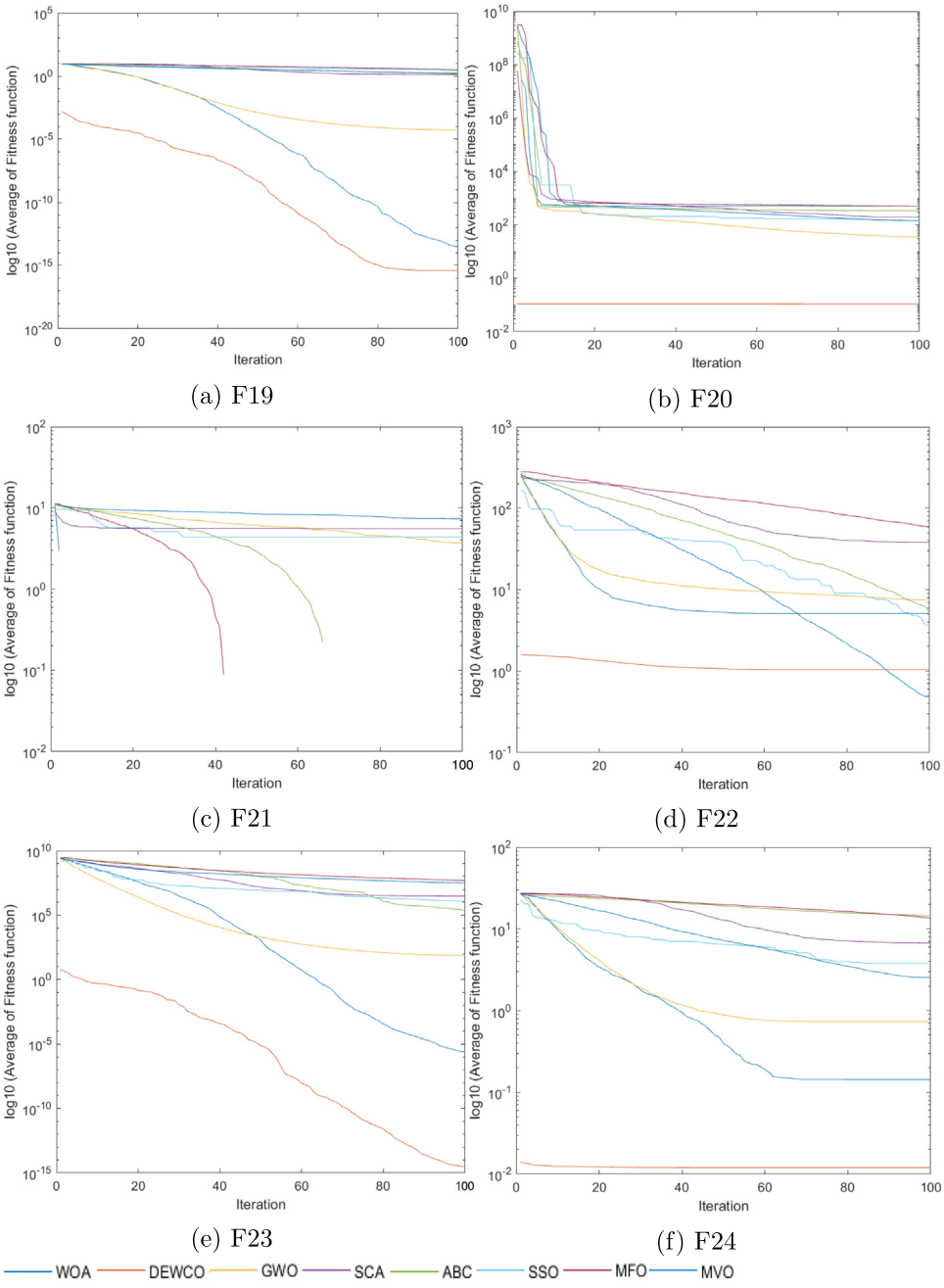


Fig. 6. The convergence curves of the algorithm along the functions F19–F24.

The main three operations of DE are repeated until the terminal criteria (usually the maximum number of iterations $MaxIter_D$ are satisfied.

2.2. Whale Optimization Algorithm

In [71], an alternative swarm algorithm was presented which called Whale Optimization Algorithm since it emulates the behavior humpback whales. Similarly to DE, a population of the solution is randomly generated first. The next step is to calculate the objective function Fit_w for each solution XW and then determine the best solution XW_b. Thereafter, the population is updated using one of the following two methods (1) encircling, and (2) bubble-net.

In the encircling method the current solution XW_i is updated using Eq. (5)

$$XW_i(t+1) = XW_b(t) - A \odot D,\ A = 2a \odot r_2 - a, \tag{5}$$

$$D = |B \odot XW_b(t) - XW_i(t)|,\ B = 2r_1 \tag{6}$$

where D_n represents the distance between the best solution $XW_b(t)$ and $XW_i(t)$ at the iteration t. The r_1 and r_2 are random number belong to the interval [0,1], while, the a is parameter where its value decreased in the interval [2, 0] during the iteration increases (i.e., $a = a - t\frac{a}{MaxIter_W}$). The symbol $\odot$ is used to represent the element-wise multiplication operation.

Moreover, in the bubble-net method the solution can be updated using (1) shrinking encircling or (2) spiral. In the case the WOA used the shrinking encircling to update solution, it used the same strategy of decreasing the value of a (i.e., $a = a - t\frac{a}{MaxIter_W}$).

In the case the WOA used the spiral strategy to update, the solution the following equation is used which emulates the helix-shaped movement of the whales around the best solution XW_b

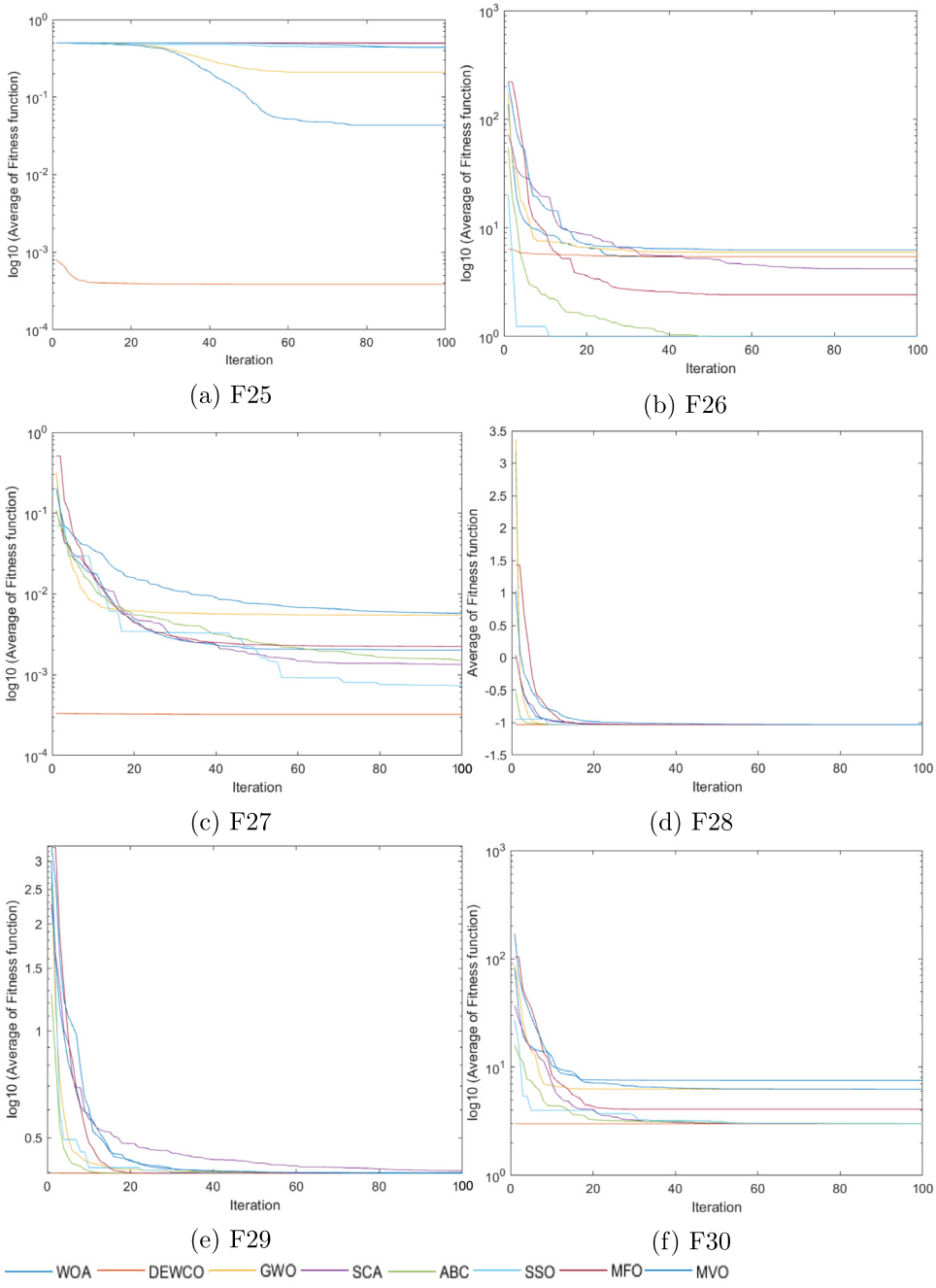


Fig. 7. The convergence curves of the algorithm along the functions F25–F30.

(also called prey) [71]:

$$XW_i(t+1) = D' \odot e^{bl} \odot \cos(2\pi l) + XW_b(t), \; D' = |XW_b(t) - XW_i(t)|, \tag{7}$$

where $l \in [-1, 1]$ is a random variable, while, the constant value b is used to specify the shape of a logarithmic spiral.

In the WOA algorithm, the solutions can updated by switching between the shrinking and spiral-shaped path as in Eq. (8) [71]:

$$XW_i(t+1) = \begin{cases} XW_b(t) - A \odot D & \text{if } r_3 \geq 0.5 \\ D' \odot e^{bl} \odot \cos(2\pi l) + XW_b(t) & \text{if } r_3 < 0.5, \end{cases} \tag{8}$$

where $r_3 \in [0, 1]$ is a random number represents the probability of swathing between the two methods (see Eqs. (5)–(7)).

In nature, whales have another ability during the search about the prey XW_b which can make random search and this can be performed by selecting a random position (XW_r) instead the best solution (XW_b) as [71]:

$$XW_i(t+1) = XW_r(t) - A \odot D \tag{9}$$

$$D = |B \odot XW_r(t) - XW_i(t)| \tag{10}$$

As per the recommendation in [71], the update of the solutions of the whale are dependent on a set of the parameters a, A, B and r_3. When the value of $r_3 > 0.5$, the solution is updated using Eq. (7). When $r_3 < 0.5$, WOA updates the solution using either Eqs. (9)–(10) or Eqs. (5)–(6) based on the $|A|$. WOA still updates the solutions until the end criterion is satisfied. After terminating the optimization process, the solution XW_b is returned as the best solution obtained by WOA. The steps of the WOA are given in Fig. 1.

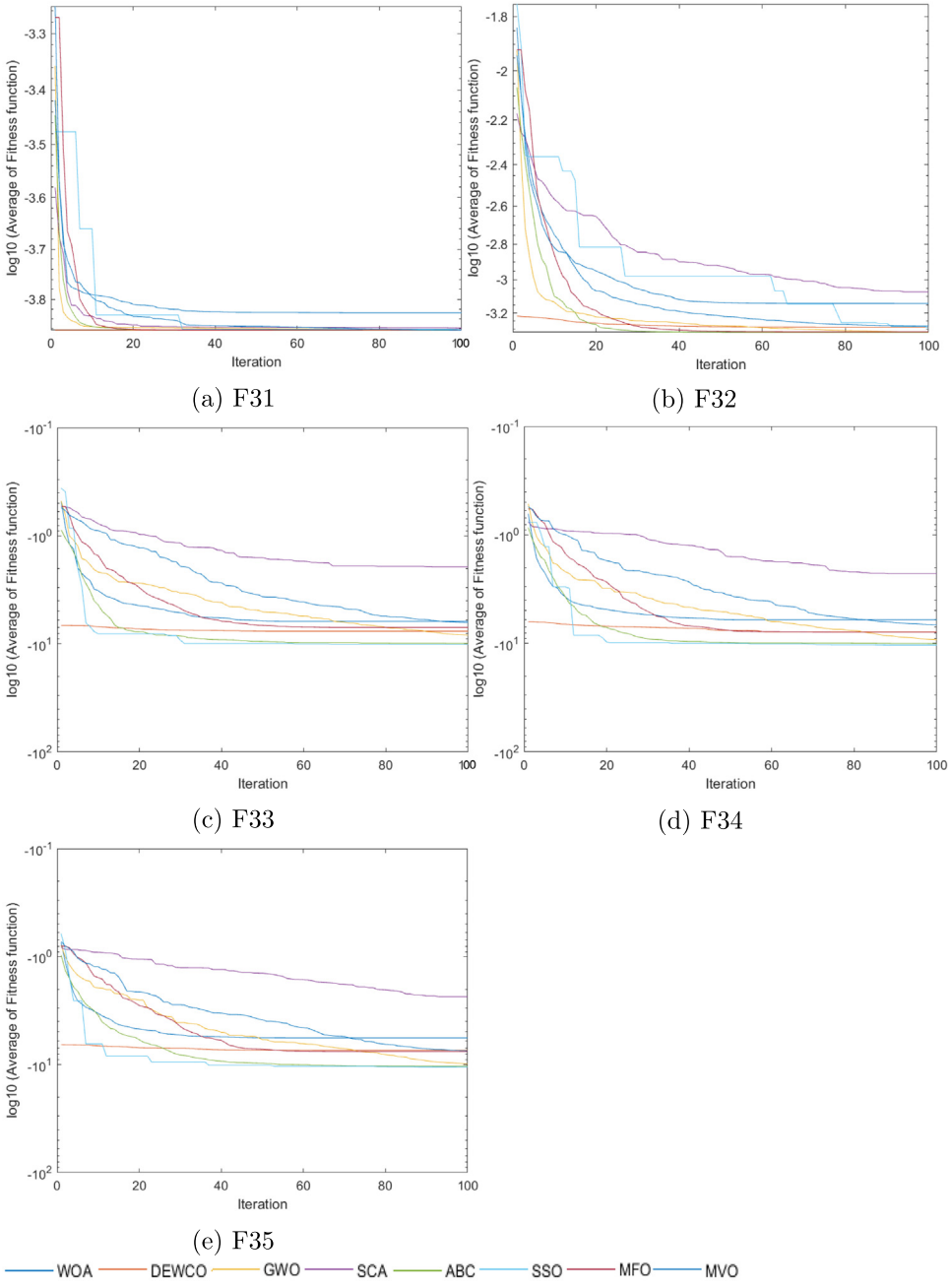


Fig. 8. The convergence curves of the algorithm along the functions F31–F35.

2.3. Chaotic maps

In this subsection, the basic concepts of the chaotic maps (CMs) are introduced. In general, most of the swarm algorithms generate a random population from different probability distributions such as Gaussian or uniform. However, this randomness might lead to slow convergence and effect on the quality of the best solution. A popular remedy to this issue is replacing random parameters with chaotic maps. Such chaotic parameters simulate strong behavior such as dynamical properties which give a high ability to the method to generate a solution with good diversity in the search space [76]. To benefit from these properties of CMs, a one-dimensional, and non-invertible maps are used to generate a set of chaotic WOA versions. The details of the ten CMs are given as in Table 1.

2.4. Opposition-based learning

In this section, the opposition-based learning (OBL) approaches are discussed [45,46]. In such methods, the current position and its opposite position are considered. The best of them is selected to be used in the next generations.

For more details, considering a real number $XW \in [LW, UW]$ (LW and UW are the lower and the upper boundaries of the given problem, respectively), the opposite number $\overline{XW}$ is computed using the following equation: [45]:

$$\overline{XW} = UW + LW - XW \tag{11}$$

The previous definition can be extended to the high dimensional solution $\overline{XW} \in R^{dim_w}$ as:

$$\overline{XW}_{ij} = UW_j + LW_j - XW_{ij}, \quad j = 1, 2, \ldots, dim_w \tag{12}$$

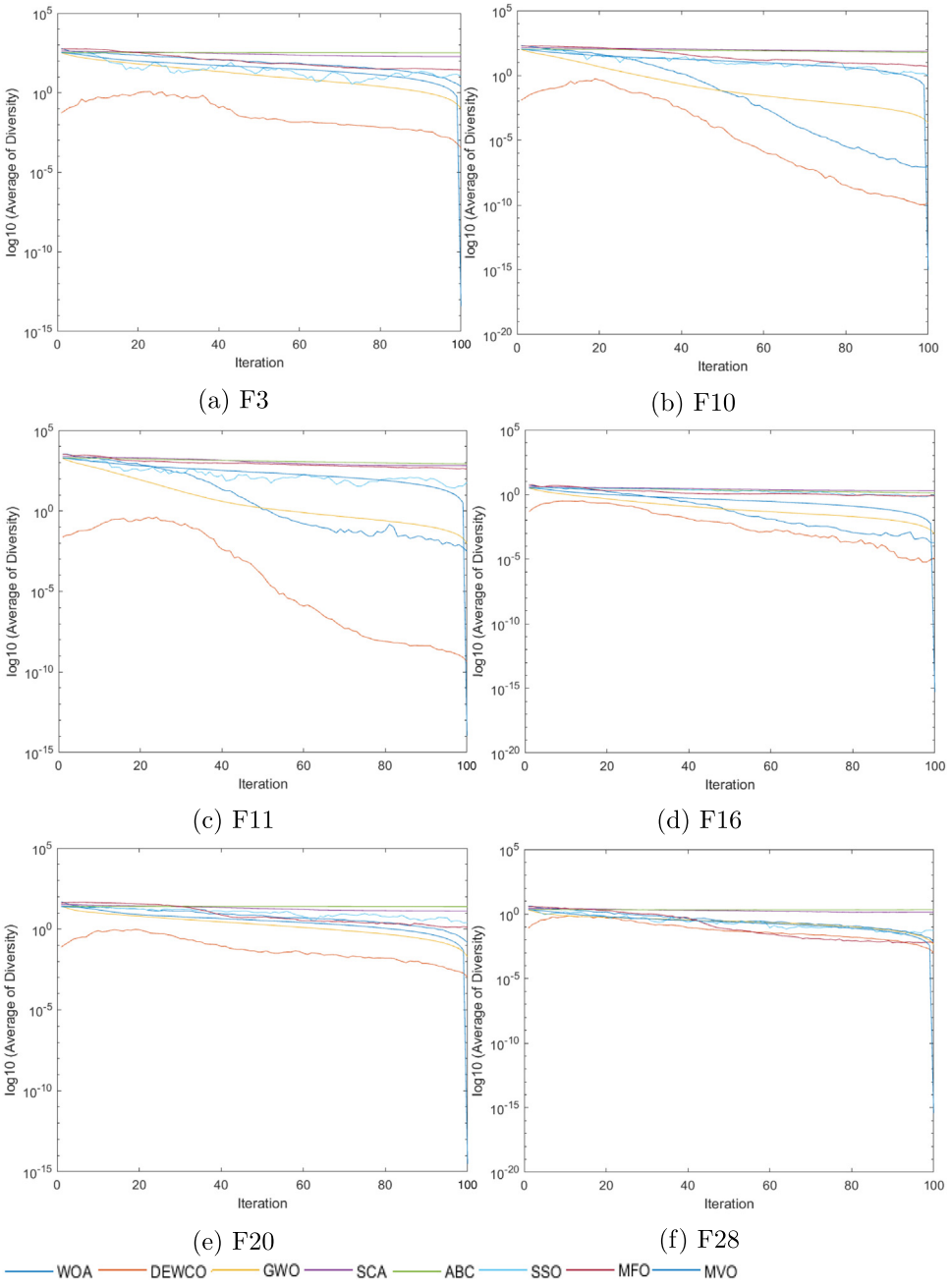


Fig. 9. The average of diversity for each algorithm along the functions F3, F10, F11, F16, F20, F28.

Thereafter, the opposite solution $\overline{XW}$ of the current solution XW is computed, then when the $Fit(\overline{XW})$ is better than $Fit(XW)$ it selected; otherwise, XW is saved. Based on the concepts of the OBL, the population contains only the best solutions from XW and $\overline{XW}$.

In general, there are different varieties of the OBL approach such as Quasi-OBL (QOBL) [77], Quasi-Reflection OBL (QROBL) [78], and super-opposite based learning (SOBL) [79]. For more details about the OBL approaches, interested readers are referred to [80]. In this study, the three OBL approaches namely QOBL, QROBL, the super opposition are used to improve the convergence of the proposed method. The definition of these approaches are given as follows:

(1) Quasi-Reflection OBL (QROBL) [78]: The Quasi-Reflected solution $\overline{XW}_{qr}$ of the solution XW is defined as a random solution which uniformly distributed between the middle point $Mid = (UW + LW)/2$ and the solution XW:

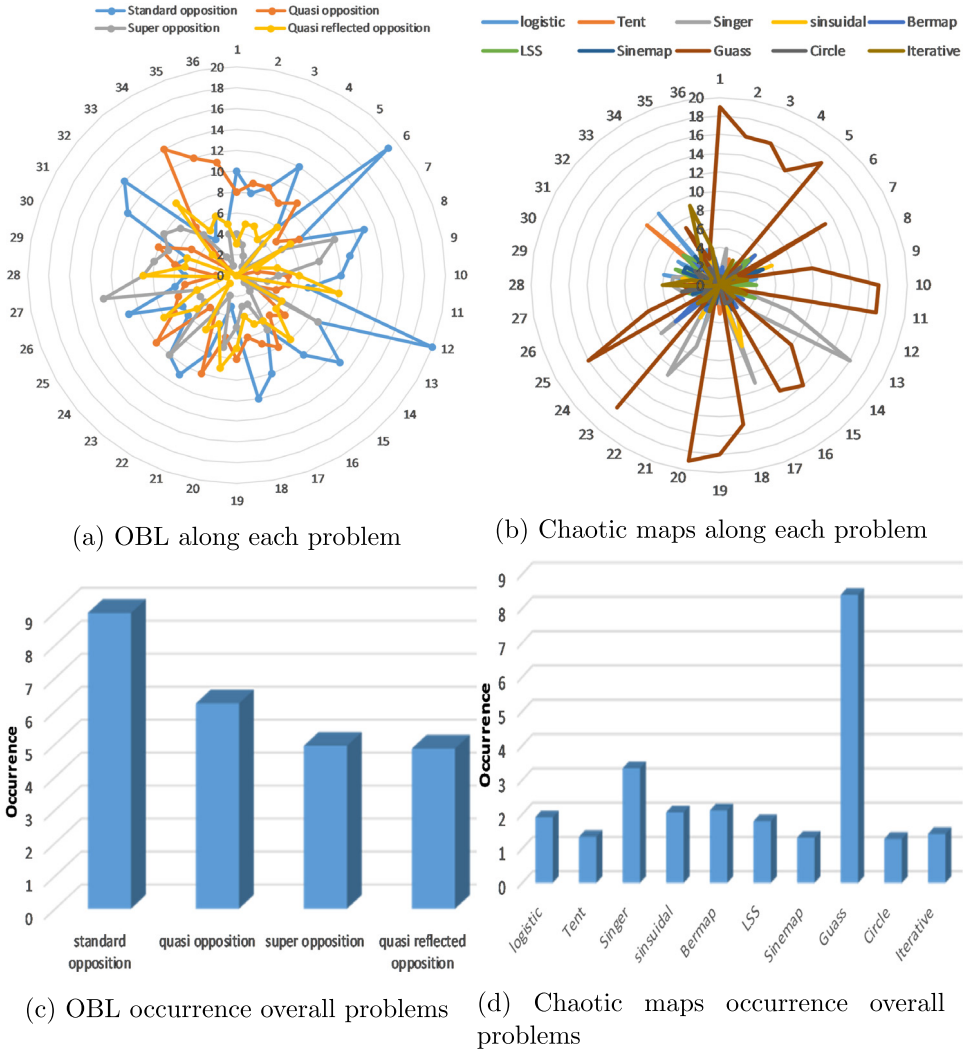
$$\overline{XW}_{qr} = UW + (Mid - XW) \times rand, \tag{13}$$

(2) The quasi-opposite solution $\overline{XW}_q$ of the solution XW is defined as a uniform random solution generated between the middle point (Mid) and the opposite solution $\overline{XW}$ as given in the following equation: [77].

$$\overline{XW}_q = Mid + (Mid - \overline{XW}) \times rand, \tag{14}$$

(3) The super-opposite solution $\overline{XW}_{so}$ of the solution XW is generated from a uniform distribution as [79]:

$$\overline{XW}_{so} = \begin{cases} \overline{XW} + (UW - \overline{XW}) \times rand, & If\ \overline{XW} > Mid \\ LW + (\overline{XW} - LW) \times rand, & otherwise \end{cases} \tag{15}$$



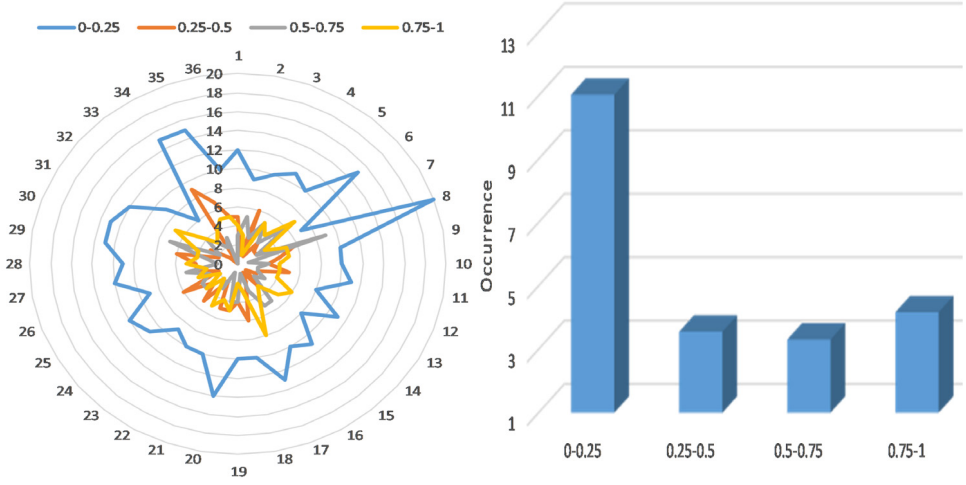
(a) OBL along each problem

(b) Chaotic maps along each problem

(c) OBL occurrence overall problems

(d) Chaotic maps occurrence overall problems

Fig. 10. The occurrence of OBL, Chaotic map, Ratio of XW and the average of occurrence of OBL methods.



(a) Ratio of XW along each problem

(b) Ratio of XW occurrence overall problems

Fig. 11. The average of the occurrence for chaotic maps and Ratio of XW.

Table 8
The results of Wilcoxon's rank sum Test using the average of the fitness value.

	WOA	GWO	SCA	ABC	SSO	MFO	MVO		WOA	GWO	SCA	ABC	SSO	MFO	MVO
F1	1	1	1	1	1	1	1	F2	1	1	1	1	1	1	1
F3	1	1	1	1	1	1	1	F4	1	1	1	1	1	1	1
F5	1	1	1	1	1	1	1	F6	1	1	1	1	1	1	1
F7	1	1	1	1	1	1	1	F8	0	1	1	1	1	1	1
F9	1	1	1	1	1	1	1	F10	1	1	1	1	1	1	1
F11	1	1	1	1	1	1	1	F12	1	1	1	1	1	1	1
F13	1	1	1	0	1	1	1	F14	1	1	1	1	1	1	1
F15	1	1	1	1	1	1	1	F16	1	1	1	1	1	1	1
F17	1	1	1	1	1	1	1	F18	1	1	1	1	1	1	1
F19	1	1	1	1	1	1	1	F20	1	1	1	1	1	1	1
F21	1	1	1	1	1	1	1	F22	1	1	1	1	1	1	1
F23	1	1	1	1	1	1	1	F24	1	1	1	1	1	1	1
F25	1	1	1	1	1	1	1	F26	0	0	0	1	1	1	0
F27	1	1	1	1	1	1	1	F28	0	1	1	0	1	1	1
F29	1	0	1	1	1	1	1	F30	0	1	1	1	1	1	1
F31	1	0	1	1	1	1	0	F32	1	1	1	1	0	1	1
F33	1	1	1	1	1	0	0	F34	1	1	1	1	1	0	0
F35	1	1	1	1	1	0	0								

Table 9
The results of increase the maximum number of iterations to 1000.

	Average		STD		Best		Worst		SR	
	WOA	DEWCO	WOA	DEWCO	WOA	DEWCO	WOA	DEWCO	WOA	DEWCO
F1	1.22E−141	**3.51E−145**	4.06E−141	**1.72E−144**	**6.14E−168**	7.48E−159	1.84E−140	**8.62E−144**	**25**	**25**
F2	1.14E−97	**1.59E−100**	5.69E−97	**5.35E−100**	2.23E−112	**2.78E−113**	2.85E−96	**1.98E−99**	**25**	**25**
F3	2.58E+04	**9.24E−03**	1.39E+04	**2.58E−02**	2.12E+03	**2.39E−07**	5.09E+04	**1.12E−01**	0	0
F4	3.88E+01	**5.84E−03**	2.94E+01	**1.51E−02**	2.59E+00	**3.60E−07**	8.31E+01	**6.77E−02**	0	0
F5	2.75E+01	**2.69E+01**	6.87E−01	**3.09E−01**	2.67E+01	**2.63E+01**	2.88E+01	**2.78E+01**	0	0
F6	1.72E−01	**2.25E−02**	2.09E−01	**1.33E−02**	1.54E−02	**8.07E−03**	8.20E−01	**6.64E−02**	0	0
F7	1.26E−03	**6.68E−04**	1.72E−03	**9.97E−04**	**2.74E−05**	3.61E−05	6.45E−03	**4.07E−03**	0	0
F8	2.05E−139	**3.01E−144**	1.03E−138	1.48E−143	**2.20E−167**	2.45E−158	5.14E−138	**7.41E−143**	**25**	**25**
F9	**5.38E−217**	1.27E−207	**0.00E+00**	**0.00E+00**	1.11E−249	**1.06E−257**	**8.23E−216**	3.17E−206	**25**	**25**
F10	**1.07E−149**	4.02E−146	**3.88E−149**	2.01E−145	**7.75E−174**	1.47E−171	**1.75E−148**	1.00E−144	**25**	**25**
F11	**−1.63E+03**	**−1.63E+03**	**0.00E+00**	**0.00E+00**	**−1.63E+03**	**−1.63E+03**	**−1.63E+03**	**−1.63E+03**	0	0
F12	**2.27E−15**	4.55E−15	**1.14E−14**	1.57E−14	**0.00E+00**	**0.00E+00**	**5.68E−14**	**5.68E−14**	**24**	**25**
F13	**4.01E−15**	4.16E−15	2.58E−15	**2.03E−15**	**8.88E−16**	**8.88E−16**	**7.99E−15**	**7.99E−15**	0	**25**
F14	5.93E−03	**0.00E+00**	2.12E−02	**0.00E+00**	**0.00E+00**	**0.00E+00**	9.28E−02	**0.00E+00**	**23**	**25**
F15	1.02E−02	**1.56E−03**	7.74E−03	**1.44E−03**	1.01E−03	**2.66E−04**	3.38E−02	**6.67E−03**	0	0
F16	4.03E−01	**2.12E−02**	2.71E−01	**1.58E−02**	9.40E−02	**4.49E−03**	1.14E+00	**6.49E−02**	0	0
F17	2.23E+00	**3.59E−01**	1.56E+00	**2.05E−01**	3.99E−01	**4.93E−02**	6.81E+00	**8.77E−01**	0	0
F18	5.20E−99	**2.78E−103**	2.60E−98	1.11E−102	3.13E−110	2.61E−114	1.30E−97	5.52E−102	**25**	**25**
F19	**0.00E+00**	1.78E−17	**0.00E+00**	8.88E−17	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.44E−16	**25**	**25**
F20	4.81E+02	**7.60E−02**	1.02E+02	**2.08E−01**	3.43E+02	**2.13E−06**	8.49E+02	**7.69E−01**	0	0
F21	−2.84E+01	**−2.72E+01**	1.15E+00	**9.31E−01**	−2.90E+01	**−2.88E+01**	**−2.52E+01**	−2.53E+01	0	0
F22	3.59E−01	**8.71E−02**	3.53E−01	**7.39E−02**	4.66E−02	**1.76E−02**	1.40E+00	**3.38E−01**	0	0
F23	1.62E−140	**3.02E−144**	7.01E−140	**1.47E−143**	9.29E−159	**1.00E−166**	3.50E−139	**7.35E−143**	**25**	**25**
F24	1.36E−01	**1.20E−02**	5.69E−02	**3.31E−02**	9.99E−02	**1.38E−82**	3.00E−01	**9.99E−02**	**2**	**25**
F25	2.56E−02	**3.89E−04**	2.33E−02	**1.94E−03**	9.72E−03	**0.00E+00**	7.82E−02	**9.72E−03**	**3**	**25**
F26	**1.83E+00**	3.93E+00	**2.03E+00**	4.19E+00	**9.98E−01**	**9.98E−01**	**1.08E+01**	1.27E+01	0	0
F27	7.18E−04	**3.18E−04**	4.16E−04	**1.43E−05**	**3.07E−04**	3.08E−04	2.29E−03	**3.66E−04**	0	0
F28	−1.03E+00	**−1.03E+00**	**1.42E−10**	9.64E−10	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	0	0
F29	**3.98E−01**	**3.98E−01**	4.96E−06	**3.76E−07**	**3.98E−01**	**3.98E−01**	**3.98E−01**	**3.98E−01**	11	**18**
F30	**3.00E+00**	**3.00E+00**	6.82E−05	**3.33E−05**	**3.00E+00**	**3.00E+00**	**3.00E+00**	**3.00E+00**	5	**9**
F31	**−3.86E+00**	**−3.86E+00**	2.35E−04	**2.65E−05**	**−3.86E+00**	**−3.86E+00**	**−3.86E+00**	−3.86E+00	4	**12**
F32	**−3.32E+00**	**−3.32E+00**	5.62E−04	**4.27E−04**	**−3.32E+00**	**−3.32E+00**	**−3.32E+00**	**−3.32E+00**	10	**11**
F33	**−9.32E+00**	−9.13E+00	**1.90E+00**	2.08E+00	**−1.02E+01**	**−1.02E+01**	**−5.06E+00**	**−5.06E+00**	0	0
F34	−8.28E+00	**−9.97E+00**	2.92E+00	**1.47E+00**	**−1.04E+01**	**−1.04E+01**	−2.77E+00	**−5.09E+00**	0	0
F35	−8.46E+00	**−9.00E+00**	2.83E+00	**2.47E+00**	**−1.05E+01**	**−1.05E+01**	−2.42E+00	**−5.13E+00**	0	0

3. Proposed hyper-heuristic

The framework of the proposed algorithm is given in Fig. 2. In general, the proposed algorithm aims to find the optimal configurations from three components namely, the chaotic map (CM), OBL method, and the ratio of the population where the OBL will apply. In order to achieve this goal, the proposed algorithm consists of two stages, the first stage is called training stage in which the DE algorithm is used to search about the optimal configuration from these components which generates the optimal initial population that will be used by WOA. Meanwhile, the second stage is called the testing stage, in which WOA updating the initial population, that coming from the first stage, using its operators. The details of these two stages are given in the following sections. Since DE is used to find an optimal configuration for WOA, it is considered as a hyper-heuristic algorithm.

3.1. Training stage

In this stage, there are two populations called DE's population (XD) and Whale's population (XW) with two maximum number of iterations one for DE algorithm ($Maxiter_D$) and one for WOA

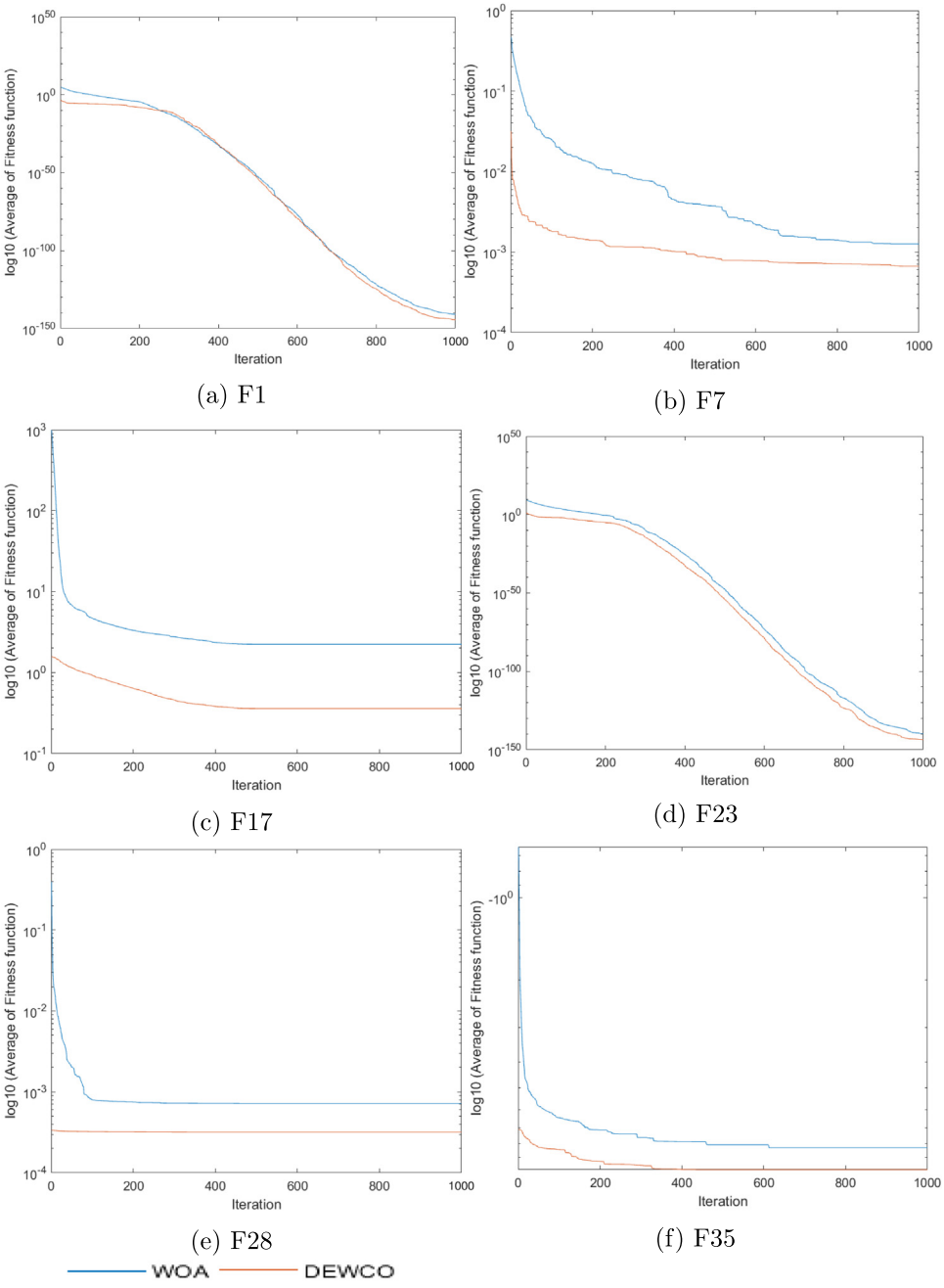


Fig. 12. The convergence curves of the WOA and DEWCO algorithms at functions F1, F7, F17, F23, F28, and F35.

($Maxiter_W$). The proposed algorithm starts by defining the set of parameters for both DE, WOA, the size and the dimension of the population $XD(XW)$ which given by $N_D(N_W)$ and $dim_D(dim_w)$, respectively. Then the population XD of size N_D and dimension $dim_D = 3$ is generated using the following equation:

$$XD_{ij} = LD_j + rand \times (UD_j - LD_j),\ i = 1, 2, \ldots, N_D, j = 1, 2, 3 \quad (16)$$

where LD_j and UD_j represent the lower and upper boundary of the jth dimension.

The first index in XD_i is the selected chaotic map (CM), and the second index represents the OBL method. The third index is the ratio of the population where the selected OBL method will be applied to it. In this study, a ten CMs, and four OBL methods are used so, $LD_j = 1, j = 1, 2$ and $UD_1 = 10$, $UD_2 = 4$. In addition the value of the $XD_{i3} \in [0, 1]$. As an example, $XD_i = [5, 2, 0.1]$ means that the fifth chaotic map is used to generate a population XW of size N_W, and the second OBL method is applied to 0.1% from the XW to compute the opposite population $\overline{XW}$. The next step is to compute the fitness function for each solution in both populations $\overline{XW}$ and XW and select the best N_W solutions, which represents the current XW, from the union of $\overline{XW}$ and XW. The process of determining the best fitness value $FitW_{best}$ and its corresponding solution XW_b is the next step. Thereafter, the values of a, A, B and p are updated and then if the $p >= 0.5$ then the current solution XW_i is updated using Eq. (7) otherwise, based on the value of $|A|$ the XW_i is updated using Eq. (9) when $|A| >= 1$ or using Eq. (5) when $|A| < 1$. After updating all the solutions in XW then check if the internal maximum number of iterations ($Maxiter_W$) is reached or not (we used it as a stopping condition).

The next step in this stage is to assign the best fitness value $FitW_{best}$ as the fitness value ($FitD_k$) for the current DE's solution $XD_k, k = 1, 2, \ldots, N_D$. After all solutions in XD are used then the

Table 10
The results of changing the $Maxiter_W$ of WOA.

	$Maxiter_W = 10$					$Maxiter_W = 15$				
	Average	Time	STD	Best	Worst	Average	Time	STD	Best	Worst
F1	**1.24E−17**	0.476	**3.07E−17**	4.57E−22	**1.47E−16**	1.40E−17	**0.449**	5.97E−17	**5.32E−24**	2.99E−16
F2	**5.37E−11**	**0.346**	**9.27E−11**	1.79E−13	**3.86E−10**	8.10E−11	0.452	3.39E−10	4.34E−13	1.70E−09
F3	2.35E−02	**0.580**	9.64E−02	**3.82E−11**	4.81E−01	1.08E−02	0.791	3.01E−02	4.77E−08	1.13E−01
F4	3.34E−03	**0.396**	4.52E−03	1.29E−05	1.99E−02	**1.71E−03**	0.412	**3.06E−03**	6.46E−06	**1.51E−02**
F5	2.81E+01	**0.342**	3.46E−01	2.74E+01	2.88E+01	2.82E+01	0.447	**2.82E−01**	**2.75E+01**	**2.87E+01**
F6	3.39E−01	**0.303**	**8.87E−02**	1.28E−01	5.00E−01	2.89E−01	0.416	1.01E−01	**1.01E−01**	5.11E−01
F7	1.77E−03	**0.365**	2.30E−03	4.02E−05	1.00E−02	**7.02E−04**	0.504	**6.89E−04**	3.28E−05	**2.47E−03**
F11	**−1.63E+03**	**0.330**	**0.00E+00**	**−1.63E+03**	**−1.63E+03**	**−1.63E+03**	0.456	**0.00E+00**	**−1.63E+03**	**−1.63E+03**
F12	7.73E−14	**0.333**	9.41E−14	**0.00E+00**	3.41E−13	5.46E−14	0.426	1.00E−13	**0.00E+00**	3.98E−13
F13	3.60E−09	**0.332**	1.04E−08	1.86E−12	4.59E−08	9.39E−10	0.445	1.66E−09	4.54E−12	7.17E−09
F14	1.60E−16	**0.345**	4.11E−16	**0.00E+00**	2.00E−15	1.02E−16	0.465	1.57E−16	**0.00E+00**	5.55E−16
F15	1.10E−02	**0.516**	5.22E−03	**5.15E−04**	**2.16E−02**	**7.28E−03**	0.680	**4.43E−03**	1.59E−03	2.32E−02
F16	9.22E−02	**0.524**	4.34E−02	**1.46E−02**	1.61E−01	8.34E−02	0.690	4.23E−02	1.59E−02	1.42E−01
F17	9.52E−16	**0.324**	2.54E−15	6.62E−21	1.07E−14	**3.48E−16**	0.443	**1.16E−15**	**3.38E−25**	**4.41E−15**
F18	2.70E−25	**0.375**	9.79E−25	5.25E−37	4.40E−24	1.36E−26	0.515	6.64E−26	**1.27E−41**	3.32E−25
Rank	2	14	3	6	4	4	1	5	8	4
	$Maxiter_W = 20$									
	Average	Time	STD	Best	Worst					
F1	2.31E−17	1.000	6.30E−17	1.35E−23	2.76E−16					
F2	1.10E−10	0.636	3.42E−10	**2.46E−14**	1.68E−09					
F3	**1.08E−03**	1.241	**2.63E−03**	2.74E−10	**1.15E−02**					
F4	3.04E−03	0.780	6.47E−03	**2.78E−06**	3.08E−02					
F5	**2.80E+01**	0.723	3.29E−01	2.76E+01	2.88E+01					
F6	**2.51E−01**	0.618	9.42E−02	1.18E−01	**4.69E−01**					
F7	1.50E−03	0.810	1.85E−03	**2.26E−05**	6.74E−03					
F11	**−1.63E+03**	0.882	**0.00E+00**	**−1.63E+03**	**−1.63E+03**					
F12	**3.87E−14**	0.769	**6.91E−14**	**0.00E+00**	**2.27E−13**					
F13	**6.55E−10**	0.790	**1.11E−09**	**2.46E−13**	**4.43E−09**					
F14	**2.22E−17**	0.614	**7.17E−17**	**0.00E+00**	**3.33E−16**					
F15	7.84E−03	0.865	7.11E−03	1.57E−03	3.66E−02					
F16	**6.71E−02**	0.887	**3.74E−02**	1.94E−02	**1.26E−01**					
F17	3.66E−16	0.553	1.78E−15	4.36E−22	8.93E−15					
F18	**7.56E−28**	0.633	**3.68E−27**	1.43E−35	**1.84E−26**					
Rank	9	0	7	8	8					

best DE's solution XD_b is determined which has the best $FitD_{best}$. Then update the solution in XD using the operators of DE algorithm (i.e., the crossover, mutation, selection) until the stopping conditions are met (here we used the maximum number of iterations ($Maxiter_D$)).

3.2. Testing stage

This stage starts by using the population XW which corresponds to the best DE's solution (XD_{best}) (i.e best configuration) as initial population. It then updates the current population XW using the operators of traditional WOA until the stop conditions are satisfied (here we used an external maximum number of iterations ($Maxiter$)). The output is the best solution to the problem.

4. Results

In order to benchmark the performance of the proposed algorithm, a set of experimental series is executed using 35 unconstrained benchmark functions taken from the standard CEC2005 test suite [81]. In this study, the number of solutions N is set to 25 (for all algorithms except DE) with a dimension equal to 30 (for all algorithms except DE), while the maximum number of iterations (Maxiter) is 150 (unless stated otherwise), which is used as the termination condition. For a fair comparison between the proposed algorithm and the other algorithms, each algorithm is executed 25 independent runs over each function. Moreover, the implementations are performed on MATLAB 2017 installed over Windows 10 (64bit) that runs on CPU Core2 Duo with 4GB RAM.

4.1. Definition of 2005 benchmark functions

In this paper, the proposed algorithm is tested using a set of different thirty-five functions form the CEC2005 benchmark of global optimization [81]. The functions of this benchmark can be classified into three different categories: (1) unimodal, (2) multimodal with variant dimension, (3) multimodal with fixed dimension.

In general, each category used to assess a specified behavior in the optimization algorithm for example, the unimodal functions are used to assess the convergence rate of the algorithm since they contain a single extreme solution in the search domain (functions F1–F10). Meanwhile, the multimodal functions are used to evaluate the ability of the algorithm to avoid the local point and reach to a global solution, and these functions contain more than an extreme solution (functions F11–F35). Table 2 gives the definition of the thirty-five functions.

4.2. Comparative algorithms

The performance of the proposed DEWCO algorithm is compared with seven state-of-the-art algorithms which including:
(1) The Whale optimization algorithm (WOA) [71]: It emulates the behavior of the whales to find the prey as discussed in Section 2.2.
(2) Grey wolf optimizer (GWO) [10]: It simulates the foraging behavior of grey wolves.
(3) Sine Cosine Algorithm (SCA) [72]: It is meta-heuristic method which used two mathematical functions called sine and cosine to update the population.
(4) Artificial Bee Colony (ABC) [73]: It represents the behavior of the bee colony as a searching method to find an optimal solution.

Table 11
The results of change the value of $MaxIter_D$.

	$MaxIter_D = 5$					$MaxIter_D = 15$				
	Average	Time	STD	Best	Worst	Average	Time	STD	Best	Worst
F1	3.26E−17	**3.010E−1**	9.36E−17	2.70E−23	3.92E−16	8.41E−19	6.328E−1	**3.27E−18**	**2.11E−26**	1.63E−17
F2	2.25E−11	**2.809E−1**	5.30E−11	1.02E−13	2.51E−10	2.27E−12	5.241E−1	4.69E−12	1.62E−15	2.08E−11
F3	2.65E−02	**3.811E−1**	7.45E−02	**7.69E−08**	3.47E−01	**1.34E−02**	2.231E+0	**2.40E−02**	4.05E−07	**8.17E−02**
F4	8.08E−03	**2.068E−1**	**1.15E−02**	1.80E−04	**3.71E−02**	**4.95E−03**	3.897E−1	1.29E−02	**2.61E−05**	6.32E−02
F5	2.82E+01	**2.149E−1**	3.16E−01	**2.77E+01**	2.88E+01	2.82E+01	4.262E−1	2.96E−01	2.78E+01	2.88E+01
F6	3.54E−01	**2.002E**-1	1.17E−01	1.90E−01	6.27E−01	*2.96E−01*	3.958E−1	**9.64E−02**	1.82E−01	**5.42E−01**
F7	1.31E−03	**2.394E−1**	1.47E−03	8.26E−05	5.72E−03	**1.19E−03**	4.887E−1	**1.23E−03**	**8.38E−06**	**4.59E−03**
F11	**−1.63E+3**	**2.114E−1**	**0.00E+0**	**−1.63E+3**	**−1.63E+3**	−1.57E+3	4.282E−1	3.00E+2	**−1.63E+3**	−1.32E+2
F12	8.41E−14	2.047E−1	1.22E−13	**0.00E+00**	4.55E−13	4.09E−14	4.171E−1	5.06E−14	**0.00E+00**	**1.14E−13**
F13	1.52E−09	**2.097E−1**	3.59E−09	8.20E−12	1.78E−08	4.29E−10	4.229E−1	1.23E−09	**2.22E−14**	6.25E−09
F14	1.33E−16	**2.239E−1**	2.52E−16	**0.00E+00**	8.88E−16	2.22E−17	4.473E−1	5.55E−17	**0.00E+00**	2.22E−16
F15	1.43E−02	**3.304E−1**	8.10E−03	1.76E−03	3.74E−02	1.39E−02	6.732E−1	7.65E−03	1.20E−03	3.55E−02
F16	1.13E−01	**3.237E−1**	**4.49E−02**	2.53E−02	**1.80E−01**	1.19E−01	6.573E−1	4.92E−02	2.79E−02	2.10E−01
F17	1.11E−15	**2.065E−1**	3.72E−15	5.60E−21	1.36E−14	4.35E−18	4.352E−1	1.75E−17	4.30E−23	8.75E−17
F18	1.83E−24	**2.388E−1**	8.60E−24	9.51E−44	4.31E−23	5.16E−26	4.709E−1	2.58E−25	9.76E−41	1.29E−24
Rank	1	15	1	1	3	3	0	3	7	3
	$MaxIter_D = 20$									
	Average	Time	STD	Best	Worst					
F1	**9.18E−20**	4.939E−1	4.20E−19	2.72E−26	**2.10E−18**					
F2	**1.31E−12**	4.904E−1	**2.67E−12**	**1.31E−15**	**1.12E−11**					
F3	5.16E−02	9.323E−1	9.66E−02	3.93E−07	3.33E−01					
F4	5.54E−03	4.793E−1	1.32E−02	1.64E−04	6.67E−02					
F5	**2.81E+01**	5.396E−1	**2.73E−01**	2.78E+01	**2.87E+01**					
F6	3.61E−01	4.712E−1	1.36E−01	7.89E−02	7.19E−01					
F7	1.22E−03	5.771E−1	1.31E−03	4.45E−05	4.69E−03					
F11	**−1.63E+03**	5.426E−1	**0.00E+00**	−1.63E+03	**−1.63E+03**					
F12	**1.14E−14**	4.766E−1	**3.28E−14**	**0.00E+00**	**1.14E−13**					
F13	**6.87E−11**	5.691E−1	**1.57E−10**	7.90E−14	**7.10E−10**					
F14	**1.33E−17**	5.874E−1	**3.68E−17**	**0.00E+00**	**1.11E−16**					
F15	**1.10E−02**	8.537E−1	**6.86E−03**	**1.00E−03**	**3.14E−02**					
F16	**1.01E−01**	8.250E−1	**4.49E−02**	**2.47E−02**	2.01E−01					
F17	**5.78E−20**	5.606E−1	**1.11E−19**	**1.54E−25**	**3.87E−19**					
F18	**5.79E−30**	6.563E−1	2.63E−29	**8.99E−44**	**1.32E−28**					
Rank	11	0	12	9	10					

(5) Social-spider Optimization (SSO) [9]: It emulates the behavior of spiders such as communication, mating and catch the prey.
(6) Moth Flame Optimization (MFO) [74]: It simulates the behavior of moths to search about the moonlight.
(7) Multi-Verse optimizer (MVO) [75]: It simulates the concepts of the White hole, Blackhole, and the Wormhole in cosmology.

The parameter values of each algorithm was set as in the original reference, while, the $N_D = 10, N_W = N, dim_D = 3, dim_W$ is set to the dimension of the tested problem. Also, $Maxiter_W = 5$, $Maxiter_D = 10$,

4.3. Performance measures

In order to assess the quality of the solutions obtained by each algorithm, a set of measures are used including the CPU time (s), average, standard deviation, worst, best of the fitness value.
(1) Average: It used to find the central value of the fitness values and it is computed as:

$$Average_F = \frac{1}{N_{run}} \sum_{i=1}^{N_{run}} FitW^i_{best} \tag{17}$$

(2) Worst value: It used to determine the largest fitness value and it computed as:

$$Worst_F = \max_{1 \le i \le N_{run}} FitW^i_{best} \tag{18}$$

(3) Best value: It used to find the smallest fitness value which calculated as:

$$Bestt_F = \min_{1 \le i \le N_{run}} FitW^i_{best} \tag{19}$$

(4) Standard deviation (STD): It used to determine deviation of the values from the central value (*Average*) and it is computed as:

$$STD_F = \sqrt{\frac{1}{N_{run} - 1} \sum_{i=1}^{N_{run}} (FitW^i_{best} - Average_F)^2}, \tag{20}$$

where N_{run} is the total number of runs and $FitW^i_{best}$ is the best fitness value obtained at the ith run.

(5) Diversity: It used to compute the variation of the solutions during the search process which defined as [82]:

$$Div = \frac{1}{N_W} \sum_{i=1}^{N_W} \left(\sqrt{\sum_{j=1}^{dim_w} (XW_{ij} - XW^{Avg}_j)^2} \right), \tag{21}$$

where XW^{Avg}_j is the jth value of the average solution $XW^{Avg} = \left[XW^{Avg}_1, XW^{Avg}_2, \ldots, XW^{Avg}_{dim_w} \right]$ and it is defined as:

$$XW^{Avg} = \left[\frac{1}{N_W} \sum_{i=1}^{N_W} XW_{i1}, \frac{1}{N_W} \sum_{i=1}^{N_W} XW_{i2}, \ldots, \frac{1}{N_W} \sum_{i=1}^{N_W} XW_{idim_w} \right] \tag{22}$$

These measures allow quantifying the performance algorithms and quantitative comparisons.

4.4. Experimental series 1: Comparison with stat-of-the-art methods

In this section, the proposed algorithm is compared with other optimization methods mentioned in Section 4.2. The comparison

Table 12
The results of change the value of N_D.

	$N_D = 5$					$N_D = 15$				
F1	6.41E−17	**0.297**	2.70E−16	4.55E−24	1.35E−15	6.91E−17	0.532	3.29E−16	6.82E−24	1.65E−15
F2	7.51E−12	**0.310**	1.14E−11	1.41E−14	4.39E−11	6.09E−11	0.536	2.56E−10	1.71E−14	1.28E−09
F3	4.60E−02	**0.480**	9.26E−02	8.90E−08	3.19E−01	3.75E−04	0.998	9.29E−04	1.04E−09	4.39E−03
F4	1.60E−02	**0.268**	2.24E−02	**1.36E−06**	7.95E−02	3.24E−03	0.526	5.05E−03	1.02E−05	2.35E−02
F5	2.82E+01	**0.289**	3.20E−01	**2.75E+01**	2.88E+01	2.80E+01	1.086	2.56E−01	2.76E+01	2.86E+01
F6	3.39E−01	**0.260**	1.40E−01	1.37E−01	5.98E−01	2.82E−01	0.518	8.20E−02	1.26E−01	4.38E−01
F7	1.49E−03	**0.327**	1.47E−03	1.37E−04	7.10E−03	1.26E−03	0.618	1.14E−03	**6.72E−05**	4.62E−03
F11	**−1.63E+03**	**0.276**	**0.00E+00**	**−1.63E+03**	**−1.63E+03**	**−1.63E+03**	0.558	**0.00E+00**	**−1.63E+03**	**−1.63E+03**
F12	5.68E−14	**0.290**	1.02E−13	**0.00E+00**	3.98E−13	4.32E−14	0.560	6.20E−14	**0.00E+00**	2.27E−13
F13	8.85E−10	**0.286**	2.09E−09	4.85E−12	9.87E−09	1.30E−09	0.660	4.09E−09	**1.29E−13**	1.63E−08
F14	4.00E−17	**0.314**	7.08E−17	**0.00E+00**	2.22E−16	4.88E−17	0.580	8.53E−17	**0.00E+00**	3.33E−16
F15	1.28E−02	**0.438**	8.80E−03	1.42E−03	3.54E−02	8.76E−03	0.934	4.36E−03	7.83E−04	1.90E−02
F16	1.07E−01	**0.514**	5.52E−02	**1.52E−02**	1.88E−01	7.78E−02	0.971	3.73E−02	1.75E−02	1.42E−01
F17	5.22E−16	**0.319**	2.52E−15	**1.81E−24**	1.26E−14	3.49E−17	0.586	1.20E−16	4.91E−24	5.89E−16
F18	3.24E−25	**0.344**	1.61E−24	6.55E−36	8.03E−24	2.84E−28	0.705	7.07E−28	3.32E−42	2.66E−27
Rank	1	15	1	7	1	1	0	1	5	1
	$N_D = 20$									
F1	**6.37E−21**	1.165	**1.74E−20**	**8.52E−26**	**7.10E−20**					
F2	**3.00E−12**	1.159	**9.17E−12**	**1.54E−15**	**4.58E−11**					
F3	**1.73E−04**	2.090	**3.18E−04**	**2.50E−10**	**1.11E−03**					
F4	**1.19E−03**	1.095	**1.18E−03**	1.44E−05	**4.25E−03**					
F5	**2.80E+01**	1.139	**1.71E−01**	2.77E+01	**2.84E+01**					
F6	**2.08E−01**	0.994	**8.41E−02**	**5.78E−02**	**3.90E−01**					
F7	**8.30E−04**	1.257	**7.14E−04**	1.19E−04	**2.67E−03**					
F11	**−1.63E+03**	1.128	**0.00E+00**	**−1.63E+03**	**−1.63E+03**					
F12	**2.73E−14**	1.118	**4.68E−14**	**0.00E+00**	**1.14E−13**					
F13	**2.12E−10**	1.111	**3.96E−10**	9.53E−13	**1.88E−09**					
F14	**1.33E−17**	1.123	**3.68E−17**	**0.00E+00**	**1.11E−16**					
F15	**5.80E−03**	1.753	**4.29E−03**	**3.28E−04**	**1.70E−02**					
F16	**7.14E−02**	1.663	**3.20E−02**	1.56E−02	**1.25E−01**					
F17	**3.69E−18**	1.114	**1.32E−17**	2.13E−24	**6.02E−17**					
F18	**5.27E−30**	1.310	**2.62E−29**	**2.96E−43**	**1.31E−28**					
Rank	15	0	15	9	15					

results are given in Tables 3–8 and Figs. 3–8. Table 3 shows that the proposed DEWCO has a better average of the fitness value the majority of test functions. For example, on the functions F1–F20, F23–F25, and F30, the proposed DEWCO achieve the first rank. It can be seen that on some functions (e.g. F28, F29, and F31) the performance of DEWCO is similar to others. However, the DEWCO cannot reach the best solution on F21, F22, F26, and F32–F35 functions. On F26, the ABC and SSO have the same performance, but ABC has the better average value at function F32. The DEWCO algorithm is very competitive in this case study.

Inspecting the results in Table 4, it can be observed that proposed DEWCO shows small fitness values on the unimodal and multimodal test functions. However, the WOA has the same value similar to the proposed DEWCO algorithm on the functions F11–F12, F19, and F26. In addition, nearly all the algorithms have the same performance at the fixed dimension multimodal function especially at F28–F35, which is due to the simplicity of such techniques. In addition, the worst fitness value achieved by each algorithm is given in Table 5. It can be concluded that the worst fitness values of the DEWCO method are better than others.

In order to evaluate the stable of the proposed DEWCO algorithm and the other algorithms the standard deviation overall number of runs is computed as in Table 6. One can observe that the SSO algorithm has the best STD value over the most functions except at six functions (i.e, F1, F11, F14, F19, and F23) where the proposed DEWCO is the better. However, the high performance of the SSO algorithm is not positive effect in all cases (functions) such as F1–F20 which has the less performance in terms of fitness value (i.e., best, worst, and average).

Table 7 shows the CPU time (s) required by each algorithm until finish all the number of iterations. It can be observed that WOA is the best algorithm on eighteen functions (F1–F4, F6–F7, F9, F12–F16, F19–F20, F22–F23, F25, F27). The MFO and SCA achieve the second and third rank on eight and seven case studies, respectively. Also, the GWO and MVO algorithms have only one best CPU time (s) at function F28, and F5, respectively.

The proposed DEWCO algorithm provides the highest time taken until the maximum number of iterations reached. However, this time is divided into two parts (1) time to find the best configuration from CMs, the ratio of OBL and the OBL methods. (2) Time computed after using the best configuration to solve the given problem. From the last two columns in Table 7, it can be seen that the time of determining the best configuration (column named Time-Train) is the main reason to increase the CPU time (s) of the proposed method. In the case of removing the time of training from the total time of DEWCO algorithm, the CPU time (s) in nearly the same of the original WOA algorithm or better than it (as in column After-Train).

Figs. 3–8 depict the convergence curves for each algorithm on each tested function. The largest effect of determining the suitable initial population to WOA is noticeable in most of the subplots. However, the DEWCO algorithm still requires improvement on some functions including F21, F22, F28, and F32–F35. When we analyze the convergence of the DEWCO algorithm on F35, it is evident that the proposed DEWCO algorithm is outperformed only by SSO, ABC, and GWO algorithms. However, it should be noted that at the beginning iterations from 1 to nearly 15, the proposed DEWCO is better than all others, which indicates the high quality of the initial population. Also, the convergence curve of the DEWCO algorithm on those functions does not change significantly after the fifteen iterations, which means that traditional WOA cannot find the optimal solution. Overall, the DEWCO algorithm shows better results than the traditional WOA.

Moreover, Fig. 9 shows the diversity of each algorithm along the following six functions F3, F10, F11, F16, F20, and F29. From this figure, it can be seen that the diversity of the proposed DEWCO

algorithm has better diversity than others at most of the six functions. However, it can be noticed that for all the six functions the MVO has better diversity overall the other algorithms after 100 iterations, but this good diversity performance for MVO not leads to improve the convergence to the global solution as concluded from previous figures and tables. Also, at the function F28 we observed that the proposed DEWCO and MFO are competitive, where the proposed DEWCO algorithm has better diversity than MFO from the beginning of iterations until nearly the 55th iteration, then the MFO has lower diversity until nearly the 95th iteration, after that the proposed DEWCO is better than MFO. Finally, this figure provides another evident about the high quality of the initial population which generated from the configuration of chaotic maps, OBL, and the ratio of the population.

4.4.1. Statistical analysis

In order to see the significance of the results in the previous experiment, a nonparametric Wilcoxon's rank sum (WRS) test is used [83]. There are two hypotheses called the null hypothesis (H) and alternative hypothesis. The former hypothesis states that there is a significant difference between the proposed algorithm and the others, whereas the latter one states that there is no significant difference between the proposed algorithm and the others. According to the statistical value, the null hypothesis is accepted if this statistical value is greater than 0.05; otherwise, the alternative hypothesis is accepted (i.e., $H = 1$)).

Table 8 shows the results of WRS test for each algorithm along each function. It can be observed that there is a significant difference between the control DEWCO algorithm and the others on the majority of test functions. However, there is no significant difference on some functions: WOA (F8, F28, and F28), GWO algorithm (F26, F29, and F31), SCA (F26), ABC (F13, and F28), SSO (F32), MFO (F33–F35), and MVO (F26, F31, F33–F35).

4.4.2. Analysis best configuration

Regarding the importance of each component in improving the performance of the WOA algorithm, the occurrence of each component among the best solution of DE during the updating process is given in this subsection. Fig. 10 shows the probability of occurrence for chaotic maps, OBL, and the ratio of whale's population (this is categorized into several intervals to compute the occurrence). This figure allows observing the contribution of each component in the best solution. Fig. 10(a) shows that there exist some functions the occurrence of one OBL method is higher than others. For example, the standard OBL method has the higher number of probability of occurrence at seventeen functions (i.e. F1, F4, F6, F8, F9, F10, F12, F13, F14, F15, F17, F18, F22, F23, F26, F31, F32). This method also reached its higher frequency at F6 and F12 which are 19 and 20, respectively. Meanwhile, the Quasi OBL is ranked second of occurrence on nine functions (F2, F5, F16, F19, F21, F24, F30, F34, F35), followed by Quasi reflected OBL with four functions (i.e., F11, F20, F25, F33) and the Super OBL method with three functions (F27, F28, F29). However, the OBL methods have the same probability of occurrence as compared to the standard OBL and Quasi OBL on F3, and F7. The average probability of occurrence is given in Fig. 10(c). It can be seen that standard OBL method is the preferred by the proposed DEWCO algorithm which has the higher probability of occurrence, followed by the quasi OBL. Note that the other two methods nearly have the same effect.

In addition, Fig. 10(b) shows the average of the probability of occurrence for each chaotic map along with each tested problem. This figure shows that the Gauss map has the largest occurrence at twenty functions followed by Singer and logistic. Fig. 10(d) shows the average of occurrence overall the tested functions, in which it provides an information that supports the fact that the chaotic Gauss map is the better chaotic map preferred by the proposed method.

It is difficult to find the optimal ratio of the population XW since this parameter is random in the interval [0,1]. Therefore, we divided the range [0,1] into four intervals (i.e. 0–0.25,0.25–0.5,0.5–0.75, and 0.75–1) and computed the occurrence of each interval. The results are given in Figs. 11(a) and 11(b). It is evident that the interval 0–0.25 has the largest occurrence which indicates that its the optimal interval. (See Fig. 11.)

4.5. Experimental series 2: Influence of increase the maximum number of iterations

In this experiment, the performance of the proposed DEWCO algorithm is compared with traditional WOA on the test functions used in previous experiments. However, only the number of iterations is changed from 100 iterations to 1000 iteration. The comparison results are given in Table 9. This table shows that the proposed DEWCO has the best value on twenty-three functions in terms of the average of the fitness value. Also, the traditional WOA has the best average at eight functions, while at the rest number of functions both of them are not significantly different. The same information can be concluded from the last column that represents the success rate (SR) of each algorithm[1]

Inspecting Table 9 and Fig. 12, it can be observed that the traditional WOA needs more iterations to reach the optimal solution. However, the proposed DEWCO still has the same performance on most of the tested functions, but it shows superior performance.

4.6. Experimental series 3: Influence of changing $MaxIter_W$ on the proposed method

In this subsection, we study the influence of the internal maximum number of iterations $MaxIter_W$ used to update the population of whale during the process of finding the optimal configuration. Table 10 shows the average, STD, Time, worst, and best fitness value at different three iterations 10, 15, and 20 with same population size N_W and the dimension of the given problem. As per the average, STD, worst, and best measures, the performance of the proposed DEWCO is increased proportionally to the number of iterations which needed to update the solutions. However, the algorithm is slower due to the time needed to find the optimal configuration.

Finally, in order to determine the optimal number of iterations that must be used to update XW, we considered that each one of the five measures has the same priority (i.e. 1/5) and computed the overall priority using the following equation:

$$Pri = \frac{Rank_{Average} + Rank_{STD} + Rank_{Time} + Rank_{Worst} + Rank_{Best}}{5} \tag{23}$$

where $Rank_M$ refers to the rank of the measure M (i.e., average, STD, time, worst, and best). Therefore, by computing Pri at each value of the maximum number of iterations, it is found that the better value of iteration may be 10 or 20 since the Pri for both of them is 6. However, if we excluded the time from our consideration then Pri of the number of iterations equal to 20 is the higher which it is 8.

[1] The SR is computed by determining how many times each algorithm achieved the following condition ($|Error| < 1E-10$) among the total number of runs.

4.7. Experimental series 4: Influence of the variants $MaxIter_D$ and N_D on the proposed method

In this section, we study the effect of the maximum number of iterations ($MaxIter_D$), which is used as a stop condition during the update the population XD, and the population size (N_D).

Table 11 shows the results of using different three $MaxIter_D$ values (i.e., 5, 15, and 20) at fixed $N_D = 10$. This table shows that the proposed DEWCO method gives the high performance at $MaxIter_D = 20$. However, according to the CPU time (s), the shortest time is obtained at $MaxIter_D = 5$. In order to determine the optimal $MaxIter_D$, Eq. (23) is used as well, in which Pri for $MaxIter_D$ at 5, 15, and 20 is equal to 4.2, 3.2, and 8.4 respectively. This indicates that the optimal $MaxIter_D$ is 20. Also, we found the minimum and the maximum of the CPU time (s) error is $1.929E-01$ and $5.512E-01$, respectively that shows the discrepancy in the execution time is not significant.

In addition, the impact of changing the population size (N_D) of DE are investigated in Table 12. Note that three value are tested (5, 15, and 20) and $MaxIter_D = 10$. This table shows that the population size of 20 lead to the best performance.

By using Eq. (23), it is found that N_D at 5, 15, and 20 has Pri equal 5, 1.6, and 10.8, respectively. These results confirm that the $N_D = 20$ is the best value.

Taken together, the results and discussions of this work showed that using a hyper-heuristic to find an optimal set of chaotic maps, OBL strategy, and the portion of the population to apply OBL is beneficial and can significantly improve the performance of the WOA algorithm. The process of generating the initial population of the WOA can significantly impacts on the performance of this algorithm. The proposed method has high ability to find the suitable initial population due to the chaotic behaviors and the OBL that explore large regions of the search space and prevent generating the same solution. This leads to more diverse solutions throughout the optimization process and better local optima avoidance, which are the main reason of the superiority of the proposed method. Hyper-heuristics are often used to tune the parameters of an algorithm, but the results of this study show that they can be used to find an optimal configuration for a meta-heuristics too. As a side effect, however, they increase the run time of the algorithm. In the case of solving the same type of problems, hyper-heuristic can be used once to find an optimal configuration. The algorithm is then applied to the problems of the same type with the optimal configuration. The results also show that DE is a very effective algorithm when we used it as the search technique in hyper-heuristic. DE is one of the most well-regarded algorithms in the literature and proven to benefit from high exploration and exploitation. Such features allowed the DE algorithm to find an optimal configuration for the WOA algorithm.

5. Conclusion

This paper presented a systematic method (hyper-heuristic) which can automatically find the optimal configuration, using DE algorithm, from three components: chaotic maps, OBL method, and the ratio of the population where the OBL must be applied to only this ratio. The main motivation of this configuration was to provide the WOA algorithm with a good initial population to increase its convergence speed. The hyper-heuristic allowed finding an optimal set of chaotic maps and OBL techniques with minimum human involvements. A set of experiments were conducted on CEC2005 benchmark functions, and the results were compared to seven state-of-arts algorithms. The experimental results showed the merits of the proposed hyper-heuristic (DEWCO) in finding an optimal configuration for WOA. This allowed to configured WOA to outperform the original WOA and other similar techniques significantly on the majority of the test function. The Wilcoxon rank sum test proved that the proposed DEWCO is statistically better than other algorithms as well.

For future works, the proposed method can be used in different applications including, Virtual machine allocation in cloud computing, multi-objective image segmentation to find the optimal threshold value, improving classification through selecting the optimal subset of features, and improving the performance of Deep learning methods. Also, other algorithms can be used to improve WOA instead of DE.

References

[1] M.Y. X. Li, Parameter estimation for chaotic systems by hybrid differential evolution algorithm and artificial bee colony algorithm, Nonlinear Dynam. 77 (1–2) (2014) 61–71.
[2] J. Neumann, C. Schnörr, G. Steidl, Combined SVM-based feature selection and classification, Mach. Learn. 61 (1–3) (2005) 129–150.
[3] J. Zelenka, Application of particle swarm optimization in job-shop scheduling problem in the recycling process, in: 11th IEEE International Symposium on Computational Intelligence and Informatics, CINTI, 2010, pp. 18–20.
[4] I. Guyon, A. Elissee, An introduction to variable and feature selection, J. Mach. Learn Res. 3 (2003) 1157–1182.
[5] S.-H. Wang, Y. Zhang, Y.-J. Li, W.-J. Jia, F.-Y. Liu, M.-M. Yang, Y.-D. Zhang, Single slice based detection for Alzheimer's disease via wavelet entropy and multilayer perceptron trained by biogeography-based optimization, Multimedia Tools Appl. (2016) 1–25.
[6] S.-H. Wang, H. Cheng, P. Phillips, Y.-D. Zhang, Multiple sclerosis identification based on fractional fourier entropy and a modified jaya algorithm, Entropy 20 (4) (2018) 254.
[7] X.-S. Yang, Nature-inspired metaheuristic algorithms, Luniver Press, 2008,
[8] D. Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning, Addison-Wesley, 1989.
[9] E. Cuevas, M. Cienfuegos, A new algorithm inspired in the behavior of the social-spider for constrained optimization, Expert Syst. Appl. 41 (2) (2014) 412–425.
[10] S. Mirjalili, S. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[11] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191.
[12] S. Saremi, S. Mirjalili, A. Lewis, Grasshopper optimisation algorithm: theory and application, Adv. Eng. Softw. 105 (2017) 30–47.
[13] F. Fausto, E. Cuevas, A. Valdivia, A. González, A global optimization algorithm inspired in the behavior of selfish herds, Biosystems 160 (2017) 39–55.
[14] M.S. Tavazoei, M. Haeri, Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms, Appl. Math. Comput. 187 (2) (2007) 1076–1085.
[15] R.C. Hilborn, Chaos and Nonlinear Dynamics: an Introduction for Scientists and Engineers, Oxford University Press on Demand, 2000.
[16] D. He, C. He, L.-G. Jiang, H.-w. Zhu, G.-r. Hu, Chaotic characteristics of a one-dimensional iterative map with infinite collapses, IEEE Trans. Circuits Syst. I 48 (7) (2001) 900–906.
[17] R.M. May, Simple mathematical models with very complicated dynamics, in: The Theory of Chaotic Attractors, Springer, 2004, pp. 85–93.
[18] Y. Li, S. Deng, D. Xiao, A novel hash algorithm construction based on chaotic neural network, Neural Comput. Appl. 20 (1) (2011) 133–141.
[19] A.G. Tomida, Matlab toolbox and GUI for analyzing one-dimensional chaotic maps, in: Computational Sciences and Its Applications, 2008. ICCSA'08. International Conference on, IEEE, 2008, pp. 321–330.
[20] R.L. Devaney, P.B. Siegel, A.J. Mallinckrodt, S. McKay, et al., A first course in chaotic dynamical systems: theory and experiment, Comput. Phys. 7 (4) (1993) 416–417.
[21] H.-O. Peitgen, H. Jürgens, D. Saupe, Chaos and Fractals: New Frontiers of Science, Springer Science & Business Media, 2006.
[22] E. Ott, Chaos in Dynamical Systems, Cambridge university press, 2002.
[23] A.L. Seyedali Mirjalili, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[24] W. Awada, T.M. Khoshgoftaar, D. Dittman, R. Wald, A. Napolitano, A review of the stability of feature selection techniques for bioinformatics data, in: Information Reuse and Integration, IRI, 2012 IEEE 13th International Conference on, IEEE, 2012, pp. 356–363.
[25] H.J. Touma, Study of the economic dispatch problem on IEEE 30-bus system using whale optimization algorithm, Int. J. Eng. Technol. Sci. 5 (1) (2016) 11–18.
[26] M. Rohani, G. Shafabakhsh, A. Haddad, E. Asnaashari, The workflow planning of construction sites using whale optimization algorithm (WOA), Turkish online J. Design Art Commun. 6 (2016) 2938–2950.

[27] A. Mostafa, A.E. Hassanien, M. Houseni, H. Hefny, Liver segmentation in MRI images based on whale optimization algorithm, Multimedia Tools Appl. 76 (23) (2017) 24931–24954.
[28] M.A. El Aziz, A.A. Ewees, A.E. Hassanien, Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation, Expert Syst. Appl. 83 (2017) 242–256.
[29] M.A. El Aziz, A.A. Ewees, A.E. Hassanien, Multi-objective whale optimization algorithm for content-based image retrieval, Multimedia Tools Appl. (2018) 1–38.
[30] M. Sharawi, H.M. Zawbaa, E. Emary, Feature selection approach based on whale optimization algorithm, in: Advanced Computational Intelligence, ICACI, 2017 Ninth International Conference on, IEEE, 2017, pp. 163–168.
[31] A.S. Reddy, M.D. Reddy, Application of whale optimization algorithm for distribution feeder reconfiguration, J. Electr. Comput. Eng. 11 (3) (2018).
[32] P.D.P. Reddy, V.V. Reddy, T.G. Manohar, Whale optimization algorithm for optimal sizing of renewable resources for loss reduction in distribution systems, Renewables 4 (1) (2017) 3.
[33] J. Nasiri, F.M. Khiyabani, A whale optimization algorithm (WOA) approach for clustering, Cogent Math. Stat. (2018) 1483565.
[34] A.N. Jadhav, N. Gomathi, WGC: Hybridization of exponential grey wolf optimizer with whale optimization for data clustering, Alex. Eng. J. (2017).
[35] M.M. Mafarja, S. Mirjalili, Hybrid whale optimization algorithm with simulated annealing for feature selection, Neurocomputing 260 (2017) 302–312.
[36] Y. Ling, Y. Zhou, Q. Luo, Lévy flight trajectory-based whale optimization algorithm for global optimization, IEEE Access 5 (99) (2017) 6168–6186.
[37] H. Hu, Y. Bai, T. Xu, A whale optimization algorithm with inertia weight, WSEAS Trans. Comput. 15 (2016) 319–326.
[38] B. Aulbach, B. Kieninger, On three definitions of chaos, Nonlinear Dyn. Syst. Theory 1 (1) (2001) 23–37.
[39] A.A. Ewees, M.A. El Aziz, A.E. Hassanien, Chaotic multi-verse optimizer-based feature selection, Neural Comput. Appl. (2017) 1–16.
[40] Z. Assarzadeh, A.R. Naghsh-Nilchi, Chaotic particle swarm optimization with mutation for classification, J. Med. Signals Sens. 5 (1) (2015) 12.
[41] L. Shen, L. Xu, R. Wei, L. Cao, Multi-swarm optimization with chaotic mapping for dynamic optimization problems, in: Computational Intelligence and Design, ISCID, 2015 8th International Symposium on, 2, IEEE, 2015, pp. 132–137.
[42] G. Kaur, S. Arora, Chaotic whale optimization algorithm, J. Comput. Design Eng. (2018).
[43] D. Prasad, A. Mukherjee, G. Shankar, V. Mukherjee, Application of chaotic whale optimisation algorithm for transient stability constrained optimal power flow, IET Sci. Meas. Technol. 11 (8) (2017) 1002–1013.
[44] D. Oliva, M.A. El Aziz, A.E. Hassanien, Parameter estimation of photovoltaic cells using an improved chaotic whale optimization algorithm, Appl. Energy 200 (2017) 141–154.
[45] H.R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in: Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on, vol. 1, IEEE, 2005, pp. 695–701.
[46] M.A. Elaziz, D. Oliva, S. Xiong, An improved opposition-based sine cosine algorithm for global optimization, Expert Syst. Appl. 90 (2017) 484–500.
[47] R.A. Ibrahim, M.A. Elaziz, S. Lu, Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization, Expert Syst. Appl. 108 (2018) 1–27.
[48] A.A. Ewees, M.A. Elaziz, E.H. Houssein, Improved grasshopper optimization algorithm using opposition-based learning, Expert Syst. Appl. (2018).
[49] H.S. Alamri, Y.A. Alsariera, Z. Kamal, et al., Opposition-based Whale Optimization Algorithm, Faculty of Computer System & Software Engineering, 2017.
[50] M.A. Elaziz, D. Oliva, Parameter estimation of solar cells diode models by an improved opposition-based whale optimization algorithm, Energy Convers. Manage. 171 (2018) 1843–1859.
[51] S. Rahnamayan, H.R. Tizhoosh, M.M. Salama, Opposition versus randomness in soft computing techniques, Appl. Soft Comput. 8 (2) (2008) 906–918.
[52] P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, in: International Conference on the Practice and Theory of Automated Timetabling, Springer, 2000, pp. 176–190.
[53] J.C. Gomez, H. Terashima -Marín, Evolutionary hyper-heuristics for tackling bi-objective 2D bin packing problems, Genet. Program. Evol. Mach. 19 (1–2) (2018) 151–181.
[54] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, S. Schulenburg, Hyper-heuristics: An emerging direction in modern search technology, in: Handbook of Metaheuristics, Springer, 2003, pp. 457–474.
[55] M. Mitra, A. Bahrololoum, H. Nezamabadi-pour, M.S. Baghshah, M. Montazeri, Cooperating of local searches based hyperheuristic approach for solving traveling salesman problem, in: IJCCI, ECTA-FCTA, 2011, pp. 329–332.
[56] M. Montazeri, H. Nezamabadi-pour, A. Bahrololoum, Exploring and exploiting effectively based hyper-heuristic approach for solving travelling salesman problem, in: The Fifth Iran Data Mining Conference, IDMC, Amirkabir University of Technology, Tehran, Iran, 2011.
[57] M. Montazeri, M.S. Baghshah, A. Niknafs, Selecting efficient features via a hyper-heuristic approach, 2016, arXiv preprint arXiv:1601.05409.
[58] A.C. Kumari, K. Srinivas, Hyper-heuristic approach for multi-objective software module clustering, J. Syst. Softw. 117 (2016) 384–401.
[59] E.K. Burke, J.D.L. Silva, E. Soubeiga, Multi-objective hyper-heuristic approaches for space allocation and timetabling, in: Metaheuristics: Progress as Real Problem Solvers, Springer, 2005, pp. 129–158.
[60] K. McClymont, E.C. Keedwell, Markov chain hyper-heuristic (MCHH): an online selective hyper-heuristic for multi-objective continuous problems, in: Proceedings of the 13th annual conference on Genetic and evolutionary computation, ACM, 2011, pp. 2003–2010.
[61] H.R. Topcuoglu, A. Ucar, L. Altin, A hyper-heuristic based framework for dynamic optimization problems, Appl. Soft Comput. 19 (2014) 236–251.
[62] K.A. Dowsland, E. Soubeiga, E. Burke, A simulated annealing based hyper-heuristic for determining shipper sizes for storage and transportation, European J. Oper. Res. 179 (3) (2007) 759–774.
[63] P. Cowling, G. Kendall, L. Han, An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem, in: Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on, vol. 2, IEEE, 2002, pp. 1185–1190.
[64] K. Storn, R. Price, Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[65] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, in: IEEE Congress on Evolutionary Computation, vol. 2, 2004, pp. 1980–1987.
[66] J. Andre, P. Siarry, T. Dognon, An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization, Adv. Eng. Softw. 32 (1) (2001) 49–60.
[67] O. Hrstka, A. Kučerová, Improvements of real coded genetic algorithms based on differential operators preventing premature convergence, Adv. Eng. Softw. 35 (3–4) (2004) 237–246.
[68] P.P. Biswas, P. Suganthan, R. Mallipeddi, G.A. Amaratunga, Optimal power flow solutions using differential evolution algorithm integrated with effective constraint handling techniques, Eng. Appl. Artif. Intell. 68 (2018) 81–100.
[69] X. Yu, X. Yu, Y. Lu, J. Sheng, Economic and emission dispatch using ensemble multi-objective differential evolution algorithm, Sustainability 10 (2) (2018) 418.
[70] T. Hamdi, J.B. Ali, V. Di Costanzo, F. Fnaiech, E. Moreau, J.-M. Ginoux, Accurate prediction of continuous blood glucose based on support vector regression and differential evolution algorithm, Biocybern. Biomed. Eng. 38 (2) (2018) 362–372.
[71] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[72] M. Seyedali, SCA: A sine cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (15) (2016) 120–133.
[73] D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm, Appl. Math. Comput. 214 (1) (2009) 108–132.
[74] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228–249.
[75] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Comput. Appl. 27 (2) (2016) 495–513.
[76] L. dos Santos Coelho, V.C. Mariani, Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization, Expert Syst. Appl. 34 (3) (2008) 1905–1913.
[77] S. Rahnamayan, H.R. Tizhoosh, M.M. Salama, Quasi-oppositional differential evolution, in: Evolutionary Computation, 2007, CEC 2007, IEEE Congress on, IEEE, 2007, pp. 2229–2236.
[78] M. Ergezer, D. Simon, D. Du, Oppositional biogeography-based optimization, in: Systems, Man and Cybernetics, 2009, SMC 2009, IEEE International Conference on, IEEE, 2009, pp. 1009–1014.
[79] M. Kaucic, A multi-start opposition-based particle swarm optimization algorithm with adaptive velocity for bound constrained global optimization, J. Global Optim. 55 (1) (2013) 165–188.
[80] S. Mahdavi, S. Rahnamayan, K. Deb, Opposition based learning: A literature review, Swarm Evol. Comput. 39 (2018) 1–23.
[81] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, in: Nanyang Tech. Univ, Singapore and KanGAL, Tech. Rep, Rep. No, Kanpur Genetic Algorithms Lab, IIT, Kanpur, India, 2005, vol. 2005.
[82] J.C. Bansal, P. Farswan, A novel disruption in biogeography-based optimization with application to optimal power flow problem, Appl. Intell. 46 (3) (2017) 590–615.
[83] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (6) (1945) 80–83.