JACET

J. ADV COMP ENG TECHNOL, 7(1) Winter 2021

A Modified Differential Evolution Algorithm with a Balanced Performance for Exploration and Exploitation Phases

Iraj Naruei[1] *, Farshid keynia*[2]

1- Department of Computer Engineering, Islamic Azad University, Kerman Branch, Kerman, IRAN

2- Department of Energy Management and Optimisation, Institute of Science and High Technology and Environmental Sciences, Graduate University of Advanced Technology, Kerman, Iran. (f.keynia@kgut.ac.ir)

Received (2020-07-24) Accepted (2021-03-18)

Abstract: *Recently optimization algorithms are proposed to find the best solution for complex engineering problems. These algorithms can search unknown and multidimensional spaces and find the optimal solution in the shortest possible time. In this paper, we present a new modified differential evolution algorithm. Optimization algorithms typically have two stages of exploration and exploitation. Exploration refers to global search, and exploitation refers to local search. We used the same differential evolution (DE) algorithm. This algorithm uses a random selection of several other search agents to update the new search agent position, which makes the search agents continually have random moves in the search space, which refers to the exploration phase. Still, there is no mechanism considered explicitly for the exploitation phase in the DE algorithm. In this paper, we have added a new formula for the exploitation phase to this algorithm and named it the Balanced Differential Evolution (BDE) algorithm. We tested the performance of the proposed algorithm on standard test functions, CEC2005 Complex, and Combined Tests Functions. We also apply the proposed algorithm to solve some real problems to demonstrate its ability to solve constraint problems. The results showed that the proposed algorithm has better performance and competitive performance than the new and novel optimization algorithms.*

Keywords: *Balanced Differential Evolution, Optimization Algorithm, Exploration and Exploitation, Constrained Search Method, Economic Dispatch Problem.*

How to cite this article:

Iraj Naruei , Farshid keynia. A Modified Differential Evolution Algorithm with a Balanced Performance for Exploration and Exploitation Phases. J. ADV COMP ENG TECHNOL, 7(1) Winter 2021 : 1-18

I. INTRODUCTION

The process of finding optimal values for certain system parameters of all possible values to maximize or minimize output is called optimization. Because common optimization techniques have problems such as local optimization stagnation and the need to derive search space, random optimization methods have become popular in the last two decades. The use of meta-algorithms significantly increases the ability to find high-quality solutions in the shortest possible time for hybrid optimization problems. The common goal of all meta-algorithms is to solve the well-known hard optimization problems [1]. Various criteria are used to classify meta-algorithms [2]. In general, meta-heuristic algorithms are divided into two categories: single-solution algorithms and population-based algorithms. Single-solution algorithms modify a solution during the search process, while in population-based algorithms, a solution population is considered. Meta-heuristic algorithms are usually inspired by the concepts of biology, animal behavior, and physics. The basic framework of all optimization algorithms is almost identical. The algorithms start with a random initial population [3], [4]. This population performs the search process in some specified iterations [5], [6]. The search process involves two stages of exploration and exploitation. At the exploration stage, the algorithm should be enriched with good design and random nature to explore different parts of the search space. The exploitation phase is usually done after the exploration phase. At this stage, the algorithm tries to focus on good solutions and improve the search operation by searching around these good solutions. A good algorithm must balance these two steps to avoid premature convergence or belated convergence. The difference between optimization algorithms is in the mechanism used to perform the search and balance in the exploration and exploitation phases. The most popular meta-heuristic algorithms include Genetic Algorithms (GA)[7], Particle Swarm Optimization (PSO)[8], Ant Colony Optimization (ACO)[9], Differential Evolution (DE)[10], and Harmony Search (HS) [11]. Genetic Algorithm (GA) is the most popular evolutionary inspiration technique that imitates the principles of Charles Darwin's theory of adaptation survival. This method involves the basic process of selection, crossover, and mutation to replace the worst solution in each generation. The PSO algorithm simulates the movement of a population of birds or groups of fish. In this algorithm, solutions are improved based on the best ones obtained by each particle so far, and the best one found by the entire population.

The ACO algorithm mimics the ants' collective behaviour in finding the shortest path from the nest to the food source. This ants behaviour has a type of group intelligence that has recently been studied by scientists. In the real world, the ants first go around randomly to find food, then return to the nest and leave a trail of Pheromone. Other ants, when they find this path, sometimes give up roaming and follow it. If they get food, they return to the nest and leave another trail next to the previous one. In other words, they reinforce the previous path. Research shows that there are many population-based optimization techniques including Firefly Algorithm (FA) [12], Bat Algorithm (BA) [13], Salp Swarm Algorithm (SSA) [14], Gray Wolf Optimization (GWO) [15], Whale Optimization Algorithm (WOA) [16], Gravitational Search Algorithm (GSA) [17], Multi-Verse Optimization (MVO) [18], Anti-Lion Optimizer (ALO) [19], Artificial electric field algorithm (AEFA) [20], Levy flight distribution (LFD) [21], Poor and rich optimization (PRO) [22], and Tunicate Swarm Algorithm (TSA) [23]. The NFL theorem proves logically that no one can propose an algorithm to solve all optimization problems [24], which means that the success of an algorithm in solving a particular set of problems does not guarantee to solve all problems of optimization with different types and different nature. In other words, all optimization techniques act the same on average, considering all optimization problems with superior performance in a subset of optimization problems. The NFL theorem allows researchers to propose new optimization algorithms or modify existing algorithms to solve a subset of problems in different domains.

Section 2 introduces the differential evolution (DE) algorithm. Section 3 proposes a formula to improve the exploitation phase of the Differential Evolution (DE) algorithm. Section 4 represents the experimental results of the test functions and real problems. Finally, Section 5 concludes the paper and discusses possible future research.

II. DIFFERENTIAL EVOLUTION (DE)

In this section, an introduction to the classical differential evolution algorithm will be presented, which will facilitate the explanation of the improved DE algorithm later on. The differential evolution (DE) algorithm was proposed by Stern and Price (1995), and proven that an evolutionary algorithm (EA) is simple but efficient. Also, the DE algorithm provides competitive performance in various fields. The DE algorithm successfully applied to finite optimization problems. The Differential Evolution (DE) algorithm uses the N individual D dimension, for example:

$X_{i,G} = \{X_{i,G}^1, ..., X_{i,G}^D\},\ i = 1,..., N$ Where N represents the number of search agents. Each dimension $X_{\min} = \{X_{\min}^1, ..., X_{\min}^D\}$ and $X_{\max} = \{X_{\max}^1, ..., X_{\max}^D\}$ is limited. Equ creates the initial population. (1) randomly in the desired space.

$$X_{i,0}^j = X_{\min}^j + rand(0,1) * (X_{\max}^j - X_{\min}^j) \quad (1)$$

Where rand is a random number with uniform distribution in the range [0, 1], the mutation operator is then used to generate the mutation vectors. can be displayed as Equ. (2):

$$V_{i,G} = X_{r1,G} + F^*(X_{r2,G} - X_{r3,G)}),\ r1 \neq r2 \neq r3 \neq i \quad (2)$$

Where $X_{r1,G}$, $X_{r2,G}$ and $X_{r3,G)}$ are randomly selected from the current population and are different from the present I individual. F is the control parameter of the mutation at different scales that are chosen randomly from the range [0.2 0.8]. After this step, the crossover operator is used to generate a new solution by Equ. (3):

$$U_{i,G}^{j} = \begin{cases} V_{i,G}^{j},\ if\ rand_j \leq CR_i\ or\ j = n_j \\ X_{i,G}^{j},\ otherwise \end{cases};\quad \begin{matrix} i = 1,2,\ldots,NP \\ j = 1,2,\ldots,D \end{matrix} \quad (3)$$

Where $rand_j$ is a random number with uniform distribution in the range [0,1] and CR_I is the crossover control parameter, which is a random number with uniform distribution in the range [0,1] and n_j is a random integer created in the range [1,D]. Finally, the selection operator selects a better individual $U_{i,G}$ and $X_{i,G}$. The better individual will survive in the next generation based on the comparison of fitness value. Equ. (4). shows the greedy selection:

$$X_{i,G+1} = \begin{cases} U_{i,G}, if\ f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G},\ \ otherwise \end{cases} \quad (4)$$

Where $f(U_{i,G})$ and $f(X_{i,G})$ are the objective function values of $U_{i,G}$ and $X_{i,G}$.

III. PROPOSED METHOD

We used the same DE algorithm. DE algorithm updates the current position of the search agent based on randomly selected operators (this makes significant exploration of the environment) and does not move toward the present Found optimized point, which is a weakness for the exploitation act. We add a formula to remove the fault of the DE algorithm. Considering that the exploitation phase usually takes place after the exploration phase, we believe half of the iterations for the basic DE state and the other half of the iterations for the exploitation phase (Our suggested formula). We are hence causing the environment to be first explored by the DE algorithm to find promising regions and then the proposed formula exploits around these bright regions. We are achieving the right balance between these two phases. The proposed method for performing the exploitation phase is as in Equ. (5):

$$U_{i,G}^{j} = 2R_1 \cos(\pi R_2) \times (gBest^j - X_{i,G}^{j}) + gBest^j \quad (5)$$

Where $U_{i,G}^{j}$, is the next position of the search agent, R_1 is a random number with a uniform distribution between range [0,1], π is the pi value 3.14, R_2 is a random number with uniform distribution in the field [-1,1], *gBest* Position is the best search agent and $X_{i,G}^{j}$ is the current position of the search agent.

$2R_1$ Makes larger random movements so that the algorithm does not get trapped in the local optimum, which means that we are also performing exploration during the exploitation phase. $cos(\pi R_2)$ Searches around the best search agent with different radius to find a better position around this search agent. Figure 1 shows how this formula works.

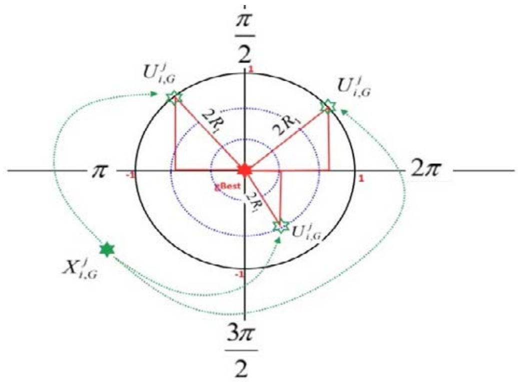

Fig. 1. Update the search agent position by the proposed formula.

The pseudocode of the proposed algorithm shown in Figure 2.

Generate the initial population of individuals
Do
  For each individual j in population
    if $\text{iter} < (\frac{\text{maxiter}}{2})$
      choose three numbers n_1, n_2, and n_3
      that is, $1 \le n_1, n_2, n_3 \le N$ with $n_1 \ne n_2 \ne n_3 \ne j$
      Generate a random integer $i_{rand} \in (1,N)$
      For each parameter i
      $V_{i,G} = X_{r1,G} + F*(X_{r2,G} - X_{r3,G})$
      $U^j_{i,G} = \left\{ \begin{array}{l} V^j_{i,G}, \text{ if } rand_j \le CR_i \text{ or } j=n_j \\ X^j_{i,G}, \text{ otherwise} \end{array} \right\}$
      End for
    else
      $U^j_{i,G} = 2R_1 \cos(\pi R_2) \times (gBest^j - X^j_{i,G}) + gBest^j$
    end if
      Replace $X^j_{i,G}$ with the child $U^j_{i,G}$ if $U^j_{i,G}$ is better
  End for
Until the termination condition is achieved

Fig. 2. Pseudo code proposed algorithm.

IV. RESULTS AND DISCUSSION

The proposed algorithm is evaluated on 19 benchmark functions, and the results are compared with popular and new population-based optimization algorithms. In general, benchmark functions can be divided into three groups: unimodal, multimodal, and composite functions. The first 13 benchmark functions are the classical test functions used by many researchers [25], [26]. From these 13 classical functions, the first seven are unimodal, and the second 6 are multimodal. The unimodal functions (f1-f7) are suitable for evaluating the exploitation phase of the algorithms because they have a global optimum and no local optimum. The multimodal functions (f8-f13) have many numbers of local optimum, and they are useful for evaluating the exploration phase and avoiding local optimal algorithms. Composite functions (f14-f19) are a combination of different uni-modal and multimodal test functions, rotating, and displacement, which are from the CEC2005 session [27]. Search space These functions are very challenging, they are very similar to real search spaces, and they are useful for evaluating algorithms in terms of balancing exploration and exploitation. The benchmark functions formula presented in Tables I to III, where Dim represents the dimensions of the function, RANGE represents the boundary of the function's search space, and FMIN is the optimal value.

TABLE I UNIMODAL BENCHMARK FUNCTIONS.

FUNCTION	DIM	RANGE	F_{MIN}
$f_1(x)=\sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
$f_2(x)=\sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
$f_3(x)=\sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j\right)^2$	30	[-100,100]	0
$f_4(x)=\max\left\{\lvert x_i \rvert, 1 \le i \le n\right\}$	30	[-100,100]	0
$f_5(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	[-30,30]	0
$f_6(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	30	[-100,100]	0
$f_7(x)=\max\left\{\lvert x_i \rvert, 1 \le i \le n\right\}$	30	[-1.28,1.28]	0

TABLE II MULTIMODAL BENCHMARK FUNCTIONS.

NCTION	DIM	RANGE	F_{MIN}
$F_8(x)=\sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[-500,500]	-418.9829*5
$F_9(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	30	[-5.12,5.12]	0
$F_{10}(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	30	[-32,32]	0
$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	30	[-600,600]	0
$F_{12}(x)=\frac{\pi}{n}\left\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right\} +\sum_{i=1}^{n} u(x_i,10,100,4)+\sum_{i=1}^{n} u(x_i,10,100,4) \quad y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\left\{\begin{array}{ll} k(x_i-a)^m & x_i>a \\ 0 & -a<x_i<a \\ k(-x_i-a)^m & x_i<-a \end{array}\right\}$	30	[-50,50]	0
$F_{13}(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)] +(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n} u(x_i,5,100,4)$	30	[-50,50]	0

TABLE III. COMPOSITE BENCHMARK FUNCTIONS. (CEC2005)

FUNCTION
$F_{14}(CF1)$: $f_1, f_2, f_3, \ldots, f_{10} = Sphere\ Function$ $[\partial_1, \partial_2, \partial_3, \ldots, \partial_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, \ldots, 5/100]$
$F_{15}(CF2)$: $f_1, f_2, f_3, \ldots, f_{10} = Griewank's\ Function$ $[\partial_1, \partial_2, \partial_3, \ldots, \partial_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, \ldots, 5/100]$
$F_{16}(CF3)$: $f_1, f_2, f_3, \ldots, f_{10} = Griewank's\ Function$ $[\partial_1, \partial_2, \partial_3, \ldots, \partial_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [1,1,1,\ldots,1]$
$F_{17}(CF4)$: $f_1, f_2 = Ackley's\ Function$ $f_3, f_4 = Rastrigin's\ Function$ $f_5, f_6 = Weierstras's\ Function$ $f_7, f_8 = Griewank's\ Function$ $f_9, f_{10} = Sphere\ Function$ $[\partial_1, \partial_2, \partial_3, \ldots, \partial_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$
$F_{18}(CF5)$: $f_1, f_2 = Rastrigin's\ Function$ $f_3, f_4 = Weierstras's\ Function$ $f_5, f_6 = Griewank's\ Function$ $f_7, f_8 = Ackley's\ Function$ $f_9, f_{10} = Sphere\ Function$ $[\partial_1, \partial_2, \partial_3, \ldots, \partial_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$
$F_{19}(CF6)$: $f_1, f_2 = Rastrigin's\ Function$ $f_3, f_4 = Weierstras's\ Function$ $f_5, f_6 = Griewank's\ Function$ $f_7, f_8 = Ackley's\ Function$ $f_9, f_{10} = Sphere\ Function$ $[\partial_1, \partial_2, \partial_3, \ldots, \partial_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [0.1*1/5, 0.2*1/5, 0.3*5/0.5, 0.4*5/0.5, 0.5*5/100, 0.6*5/100, 0.7*5/32, 0.8*5/32, 0.9*5/100, 1*5/100]$
DIM= 10 RANGE= [-5,5] FMIN= 0

The values of the parameters of the algorithms compared to the proposed algorithm are presented in Table IV.

TABLE IV. INITIAL VALUES FOR THE CONTROLLING PARAMETERS OF ALGORITHMS.

algorithm	parameter	value
GA	Type Selection Crossover Mutation	Real coded Roulette wheel Single point 0.8 0.3
PSO	Topology Cognitive and social constants Inertial weight	Fully connected C1=2 , c2=2 Linearly decreases from 0.9 to 0.4
FA	Alpha, beta, and gamma	0.5, 0.2, 1
ACO	Intensification Factor Deviation-Distance Ratio factor	0.5 1 0.8
BA	Loudness (A), pulse rate (r)	0.5, 0.5
GSA	norm, Rpower, alpha, and G0	2, 1, 20, 100
DE	Crossover probability Differential weight	0.2 [0.2-0.8]
HS	Number of New Harmonies HMCR Pitch Adjustment Rate Fret Width Damp Ratio	20 0.9 0.1 0.995
BDE, WDE, WOA, GWO, SSA, ALO, MVO,	There are no parameters other than the common parameters (i.e., the number of iterations, N, and D)	

1. Exploitation analysis of the proposed BDE algorithm

In this section, we tested the proposed algorithm on seven unimodal functions used to measure the exploitation of algorithms. The proposed algorithm compared with the basic DE algorithm and two new algorithms called Weighted Differential Evolution[28] and Bernstein-search differential evolution [29]. As can be seen from the results of Table V, the proposed algorithm in F1, F2, F5, and F6 functions yields much better results than the other three algorithms. It is noteworthy that the proposed algorithm performs better than the basic DE algorithm in all 7 test functions, which indicates that the formula added to the DE algorithm improves the exploitation phase.

TABLE V. RESULTS FOR THE UNIMODAL BENCHMARK FUNCTIONS

Function	Algorithm	DE	WDE	BSD	Proposed
F1	min	2.4504e-04	4.7952e+03	0.0254	**3.6737e-13**
	max	8.2719e-04	1.2544e+04	0.1654	**1.2511e-10**
	avg	5.2748e-04	9.9787e+03	0.0701	**1.6862e-11**
	std	1.4245e-04	1.6565e+03	0.0343	**2.4829e-11**
F2	min	0.0018	30.2875	0.0176	**6.1865e-09**
	max	0.0037	50.2461	0.0763	**5.7654e-05**
	avg	0.0026	44.5314	0.0450	**2.0857e-06**
	std	4.9552e-04	4.3281	0.0139	**1.0499e-05**
F3	min	1.9105e+04	1.5003e+04	**486.2315**	1.0950e+03
	max	3.9876e+04	3.2023e+04	**1.5224e+03**	4.2786e+04
	avg	3.0893e+04	2.5231e+04	**982.6995**	1.6203e+04
	std	4.7921e+03	3.9261e+03	**260.5432**	9.8637e+03
F4	min	10.5911	45.5220	**1.7824**	5.3869
	max	16.5127	63.8297	**4.3840**	27.3616
	avg	13.2160	57.3719	**2.8596**	12.3617
	std	1.4473	3.6722	**0.6606**	4.7869
F5	min	83.7544	3.8437e+06	34.9573	**11.0130**
	max	244.9160	1.5796e+07	221.3012	**149.3052**
	avg	157.9563	1.0006e+07	119.4331	**57.0792**
	std	46.7152	2.6483e+06	45.6118	**36.7103**
F6	min	2.8240e-04	6.9130e+03	0.0155	**4.7905e-14**
	max	0.0010	1.3833e+04	0.1379	**2.2424e-10**
	avg	5.4945e-04	1.0144e+04	0.0511	**2.1664e-11**
	std	1.9155e-04	1.7063e+03	0.0300	**4.7440e-11**
F7	min	0.0323	3.4729	**0.0122**	0.0142
	max	0.0728	8.1122	**0.0528**	0.0919
	avg	0.0540	5.2874	**0.0300**	0.0429
	std	0.0134	1.1566	**0.0112**	0.0200

2. Exploration analysis of the proposed BDE algorithm

In this section, we tested the proposed algorithm on six multimodal test functions to evaluate the performance of the exploration phase of the algorithm. The proposed algorithm was compared with the basic DE algorithm and two new algorithms called Weighted Differential Evolution and Bernstein-search differential evolution. Table VI show these results. The proposed algorithm performs better in F10 and F11 functions than other algorithms. Hence to minimize or maximize the objective functions in optimization, an algorithm that can obtain the minimum value in minimization and maximum value in the maximization of the objective function would be better. The proposed algorithm in F12 and F13 test functions has the lowest value compared to other algorithms, which indicates that the proposed algorithm has an excellent ability to find the optimal value. The results of this table show that adding the proposed formula to the DE algorithm not only had little effect on the results of the test functions but also performed better in some cases. The diagram of some of the unimodal and multimodal test functions shown in Figure 3.

TABLE VI. RESULTS FOR THE MULTIMODAL BENCHMARK FUNCTIONS.

Function	Algorithm	DE	WDE	BSD	Proposed
F8	min	-9.8668e+03	-8.4058e+03	-9.2126e+03	-10.8668e+04
	max	-8.9599e+03	-7.3124e+03	-7.7869e+03	-8.8453e+03
	avg	-9.5420e+03	-7.7470e+03	-8.3110e+03	-10.6132e+04
	std	358.7499	236.9287	300.8447	1673.5341
F9	min	69.3267	160.4797	**28.2491**	76.3937
	max	99.9105	200.6283	**53.9470**	137.3118
	avg	87.7591	180.5386	**43.6630**	116.3648
	std	7.2161	10.4880	**6.2818**	13.1324
F10	min	0.0047	15.5957	0.0309	**1.5598e-07**
	max	0.0082	17.2140	0.1187	**4.2016e-06**
	avg	0.0063	16.5010	0.0675	**9.5691e-07**
	std	0.0010	0.4934	0.0187	**8.1089e-07**
F11	min	0.0012	62.8098	0.0485	**2.1294e-13**
	max	**0.0230**	105.6460	0.3091	**0.0443**
	avg	**0.0067**	86.4470	0.1404	**0.0073**
	std	**0.0061**	12.0908	0.0684	**0.0110**
F12	min	3.7585e-05	3.7904e+05	1.0733e-04	**1.3508e-14**
	max	1.7133e-04	1.2279e+07	0.0026	**0.1037**
	avg	7.9758e-05	4.3868e+06	7.6405e-04	**0.0035**
	std	4.0127e-05	2.7323e+06	6.0704e-04	**0.0189**
F13	min	1.8386e-04	6.7397e+06	8.7938e-04	**5.5576e-13**
	max	**5.9877e-04**	3.6276e+07	0.0557	0.0989
	avg	**3.0639e-04**	2.2674e+07	0.0162	0.0110
	std	**1.0023e-04**	7.4738e+06	0.0131	0.0202

3. Balance Analysis of Exploration and Exploitation of the Proposed Algorithm

What is essential in designing optimization algorithms is that the algorithm must create a right balance between the two phases of exploration and exploitation. Suppose the algorithm fails to achieve balance well. In that case, the algorithm either involves premature convergence because it failed to perform the exploration well or involves late convergence because it failed to operate well. Therefore, newly introduced or improved algorithms required to test in terms of balance the two phases of exploration and exploitation. To evaluate the balance between exploration and exploitation of the proposed algorithm, we used CEC2005 benchmark functions, which are composite and complex. These functions have many local optimal, and the algorithm may get stuck in local optimal.

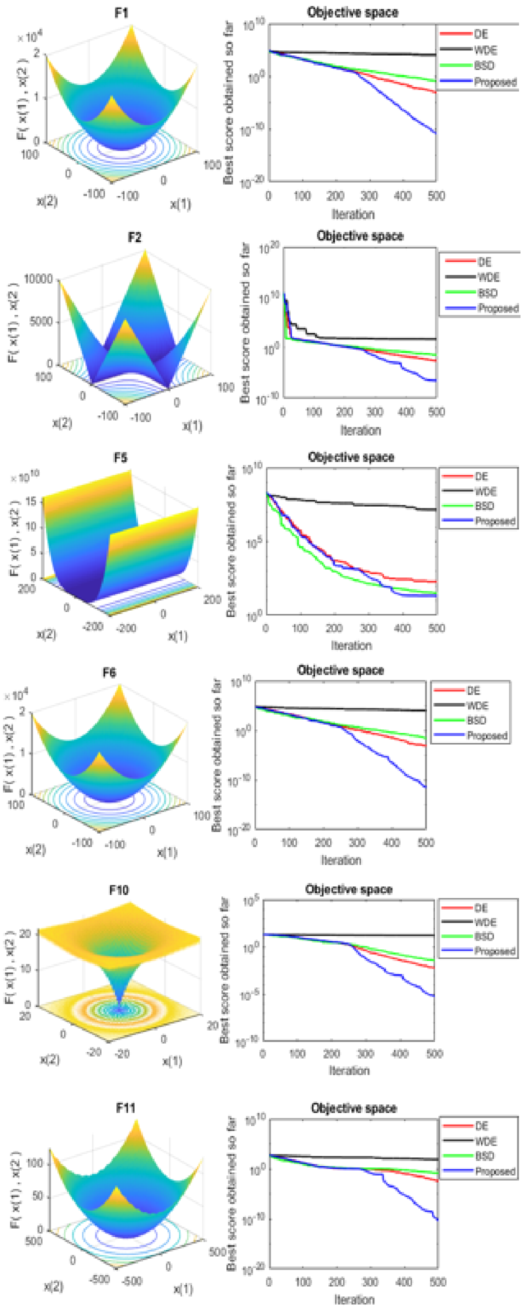

Fig. 3. Results of some test functions.

We compared the proposed algorithm with popular optimization algorithms such as PSO, GA, ACO, GSA, FA, DE, HS, and new optimization algorithms such as BA, GWO, ALO, MVO, WOA, and SSA. The results of Table VII show that in terms of mean and standard deviation, the performance of the proposed algorithm is better than all the compared algorithms. One of the most famous charts, which shows many descriptive statistics indicators of data, is the boxplot [30]. To prove this claim, we showed the boxplots of each six functions in Figure 4.

The boxplot illustrates well the domain of variations of different runs. As shown in Figure 4, the proposed algorithm has the least domain of variation in all six functions, which indicates that the results of the proposed algorithm are not random, and the results are reliable. It's noteworthy that some algorithms have been removed from the boxplot due to their poor performance to see the results of other algorithms better.

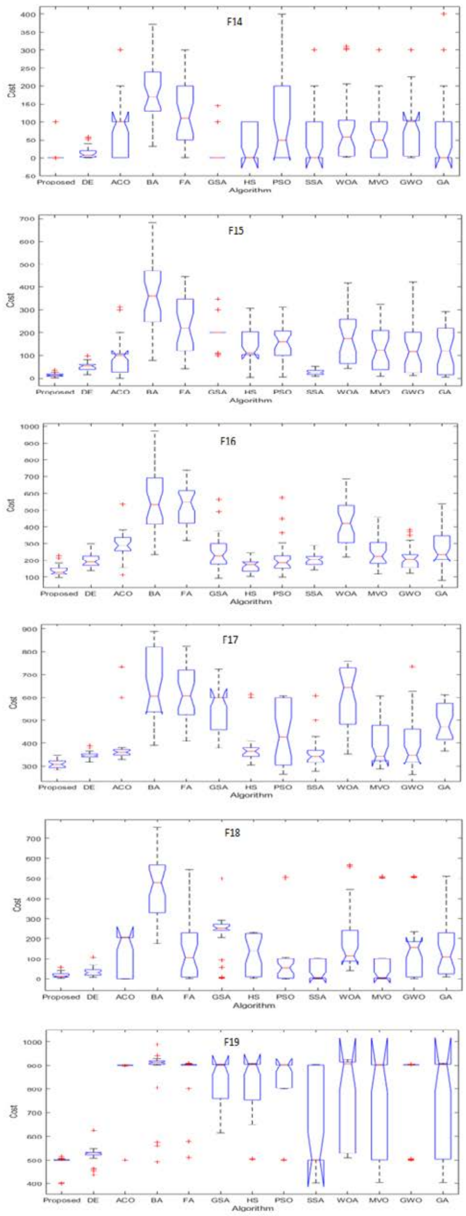

Fig. 4. Boxplot of CEC2005 benchmark functions.

TABLE. VII. RESULTS FOR THE COMPOSITE AND COMPLEX FUNCTIONS.

Algorithm / Function		PSO	ACO	GSA	FA	GA	BA	HS	DE	WOA	SSA	MVO	ALO	GWO	Proposed
F14	min	1.5875e-09	0	1.5926e-17	2.6930e-13	0.0029	32.6261	0.0109	0.0039	1.1489	1.0305e-10	0.0035	397.4413	0.0272	**0**
	avg	93.3333	76.6667	18.1826	117.4096	80.0191	188.9511	33.4264	14.8035	91.2377	60.0000	83.3396	599.7261	75.1945	**3.3333**
	max	400.0000	300	145.4773	300.0000	400.0006	372.5525	100.0512	58.4321	310.9591	300.0000	300.0110	754.5454	300.0179	**100**
	std	117.2481	85.8360	42.0367	78.3665	109.5401	82.1159	47.9039	17.7941	106.5350	85.5006	101.9914	84.9694	76.3767	**18.2574**
F15	min	5.0769	**0**	100.0000	40.5689	4.5953	76.9688	3.4771	14.1667	42.0367	7.6355	7.8929	367.6698	10.8568	1.3913
	avg	154.6440	98.1215	208.6797	229.4857	128.2596	348.9973	143.0915	52.0919	179.2558	26.1350	134.1337	704.9707	136.5402	**12.9551**
	max	312.1533	312.1533	346.3061	446.6643	292.2114	681.8824	305.4230	97.0981	418.0740	50.3143	323.9771	972.8750	422.0006	**34.2681**
	std	98.5157	94.0449	54.5958	123.6389	103.2858	160.5098	85.0365	18.9692	114.4808	12.2996	89.7563	148.6437	118.4057	**7.3662**
F16	min	98.6116	113.2252	92.9694	316.5141	**81.3590**	232.7017	102.4075	138.1543	219.6151	141.9804	119.5522	734.1056	123.8327	96.4661
	avg	213.5193	294.2018	244.5110	530.0246	266.3518	549.3272	168.5382	197.0508	426.1589	205.1376	241.8100	1.0262e+03	211.7950	**138.9825**
	max	573.0122	532.8344	564.1437	738.5087	536.0576	971.5559	245.1473	300.0453	687.6983	288.9038	458.8793	1.3065e+03	383.8393	**227.2639**
	std	100.9759	79.1417	108.6843	122.4081	108.6906	183.9312	37.2124	37.2512	133.0421	39.0231	87.8533	152.5953	71.1985	**31.6866**
F17	min	265.2150	329.1214	379.3369	410.3104	365.7324	391.2444	304.2207	319.0547	354.1240	276.6679	289.1526	858.1156	**263.7557**	283.0750
	avg	440.9509	386.5058	545.9930	609.2554	487.3977	653.5975	375.9608	345.7826	611.2633	354.9927	405.1630	1.0296e+03	405.1075	**308.8969**
	max	607.5132	732.6510	722.3274	824.6867	611.5647	889.3420	613.0268	390.1966	758.4532	607.2507	607.1968	1.2858e+03	734.9262	**348.5650**
	std	135.2587	90.6269	85.4338	120.4006	82.8341	162.5537	68.0334	15.5664	135.2981	66.2669	113.5859	100.3885	130.4053	**18.0222**
F18	min	2.8984e-05	**0**	4.3055	2.7234	10.2664	175.8204	3.9995	8.1057	41.5599	1.6230	2.5874	271.4260	2.2240	3.9596
	avg	92.9367	122.8017	228.3092	153.9865	134.9226	452.4372	122.8914	34.5088	186.8345	31.3749	91.4417	705.6995	153.2369	**17.9327**
	max	508.8939	205.1854	500	545.1053	511.6341	755.9447	230.4270	108.2184	569.3977	103.9091	509.8884	982.0372	510.3997	**55.5628**
	std	146.5752	101.0622	101.5676	162.4923	115.1310	154.8680	96.1000	20.9960	155.0380	43.1760	169.9353	218.7547	161.7342	**12.6775**
F19	min	500.0002	500	613.1760	511.2597	403.1994	491.0319	504.1762	437.0618	509.6863	401.6789	403.9472	801.2347	500.9246	**400.0000**
	avg	825.4653	848.1431	829.1813	876.4646	778.0474	876.4716	825.7530	522.5745	795.3360	635.3327	725.6787	960.7075	836.8471	**495.6028**
	max	903.0465	902.9593	904.9380	908.9796	908.9500	987.6315	907.9610	626.0057	924.3305	903.9652	903.4403	1.0114e+03	905.4697	**514.1832**
	std	150.9544	138.8490	101.1551	92.6154	191.9231	117.0116	122.0777	33.8458	173.6248	188.9743	206.5175	40.2025	152.0038	**25.8704**

4. Convergence analysis of the proposed algorithm

The movement of the search agents in the early stages of the optimization algorithms is sudden to explore the entire search area, and gradually the movement becomes slower to perform exploitation [31]. The convergence behaviour of the proposed algorithm is shown in Fig. 3, where the search history and the path of the first search agent are plotted in the first dimension. The second column of Fig. 3 shows the search history of all search agents. Column three shows convergence, at the beginning of the algorithm are sudden movements which gradually converge at one point. The fourth column of Fig. 5 shows average fitness for all search agents in each iteration. The fifth column is the convergence curve that represents the best value in each iteration.

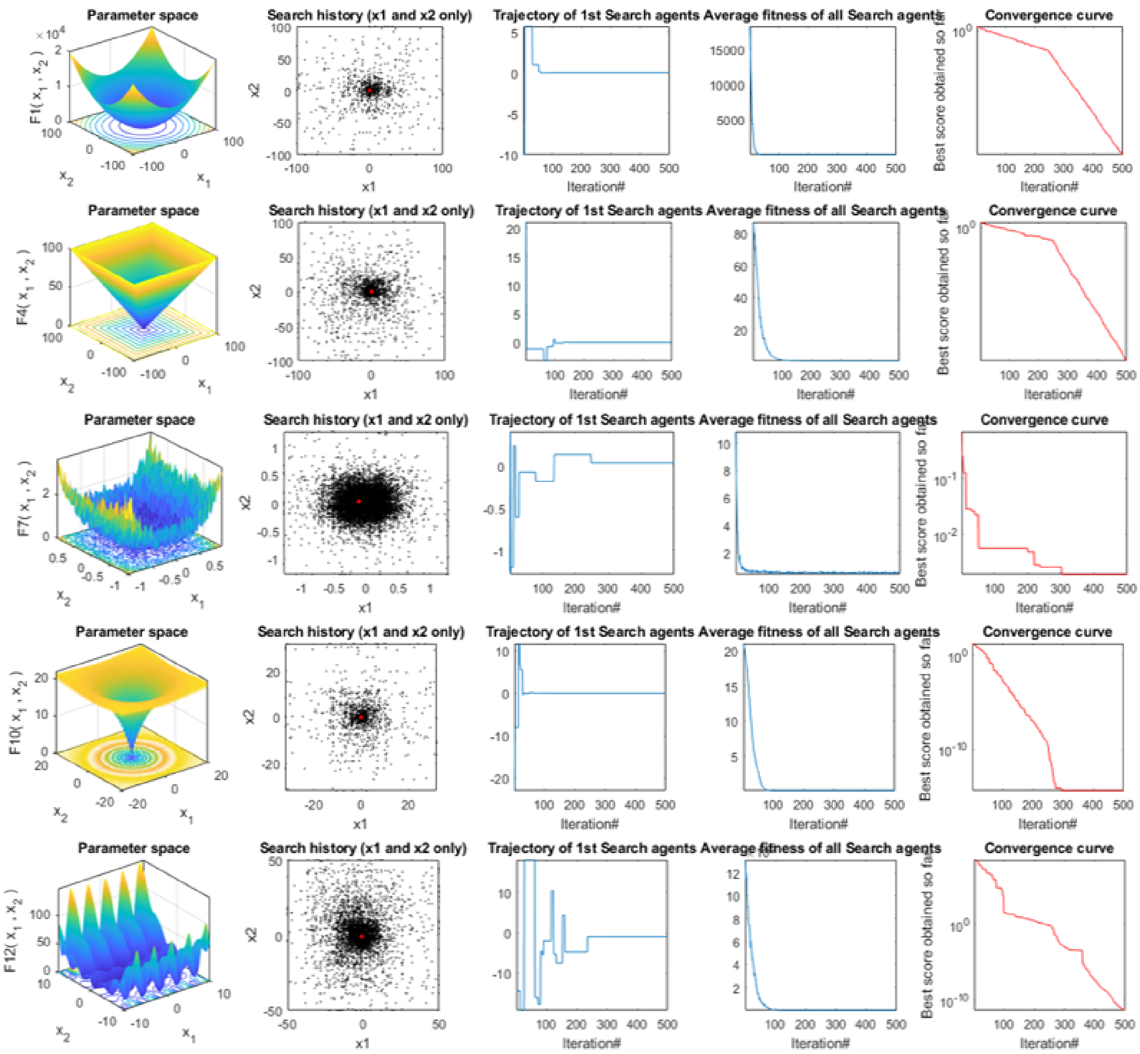

Fig. 5. Search history and trajectory of the first search agent in the first dimension.

5. Performance of proposed algorithm on constrained problems

In this section, there are three constrained problems in engineering design that have been used by many researchers: Tension/compression spring, pressure vessel design, and 3-bar terrace design. These problems have several constraints on equality and inequality. The algorithm should be able to optimize the constrained issues as well.

1) Tension/Compression spring design problem

As shown in Fig. 6. The main goal of this engineering design problem is to minimize the weight of the spring involving three decision variables which are wire diameter (d), mean coil diameter (D), and some active coils (N) [32]. This problem is subjected to three inequality constraints and an objective function given in Equ. (6).

$$\begin{aligned}
&\vec{x} = [x_1\, x_2\, x_3] = [dDN],\\
&f(\vec{x}) = (x_3 + 2)x_2 x_1^2,\\
&g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \le 0,\\
&g_2(\vec{x}) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} \le 0,\\
&g_3(\vec{x}) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0,\\
&g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \le 0,\\
&0.05 \le x_1 \le 2.00,\\
&0.25 \le x_2 \le 1.30,\\
&2.00 \le x_3 \le 15.0
\end{aligned} \tag{6}$$

This problem has solved with many mathematical and meta-heuristic approaches [34]. The results of comparing the proposed algorithm with different optimization methods with the same penalty function presented in Table VIII. The proposed algorithm was able to find the best solution with the least weight. The boxplot of these results shown in Fig. 7.

Fig. 7 shows that the proposed algorithm domain of changes is low, and the results of Table VIII are reliable.

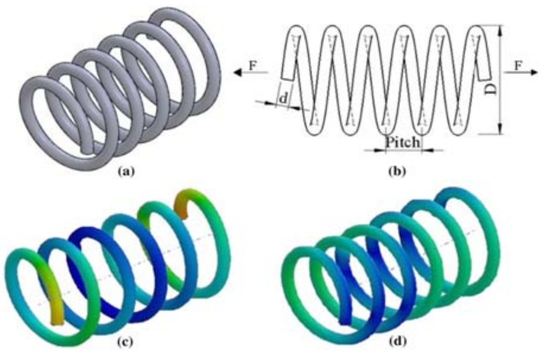

Fig. 6. a 3D view of the spring, b 2D view of the spring, c displacement heat map, d stress heat map [33].

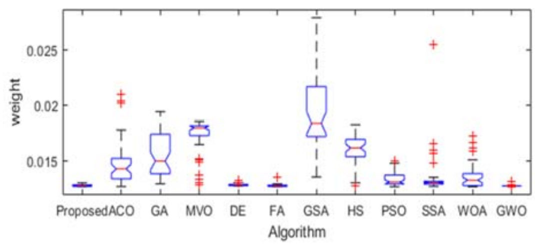

Fig. 7. Boxplot of optimization algorithms for Tension/compression spring

TABLE VIII RESULTS FOR TENSION/COMPRESSION SPRING

Algorithms	d	D	N	weight
Proposed	**0.051674240269161**	**0.356361294682208**	**11.309893654220970**	**0.012665236795762**
WOA	0.051570755371899	0.353878269149070	11.457442622631019	0.012665531420461
PSO	0.051520372987135	0.352673093103354	11.530090557248791	0.012665753370316
FA	0.051617750492592	0.355001881898848	11.394109546335027	0.012669009152984
BA	0.051808745345115	0.359551974034547	11.129493328259862	0.012671143067769
GWO	0.051279143996523	0.346845349283398	11.896862969127122	0.012674597731402
DE	0.052364855310668	0.372334483409713	10.457666318851194	0.012718906350874
ACO	0.053691799675533	0.406838948323859	8.859270871177701	0.012736177748293
MVO	0.050000000000000	0.317175182071943	14.063890263180618	0.012737668297520
SSA	0.050000000000000	0.313654356100474	14.539845769754217	0.012840126986158
GA	0.056478105859860	0.483118711787788	6.480103511901926	0.013068184527540
GSA	0.056101588408330	0.465552293653536	7.240154020410609	0.013539355647122
HS	0.059118720699916	0.561133954332981	4.964410699642398	0.013658436187036
ALO	0.064732195539379	0.633802689219889	8.412993605043161	0.027654789299257

2) Pressure vessel design

The objective of this problem is to minimize the total cost consisting of material, forming, and welding of a cylindrical vessel as in Fig. 8. Vessels both ends are capped, and the head has a hemispherical shape. There are four variables in this problem:

The thickness of the shell (Ts).

The thickness of the head (Th).

Inner radius (R).

Length of the cylindrical section without considering the head (L).

This problem is subject to four constraints. The formulation form of these constraints and problems in Equ. (7):

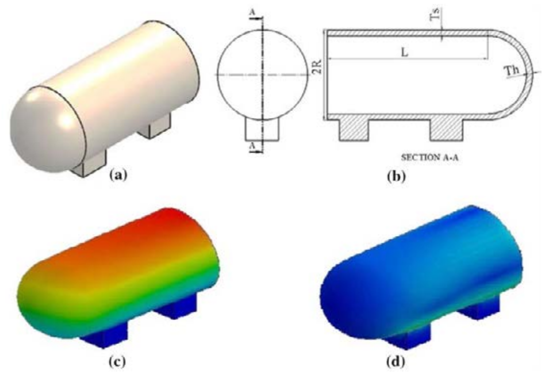

Fig. 8. (a) 3D shape of the pressure vessel, (b) 2D shape of the pressure vessel, (c) displacement heat map, (d) stress heat map [33].

$$
\begin{aligned}
&\vec{x} = [x_1\ x_2\ x_3 x_4] = [T_s T_h R L],\\
&f(\vec{x}) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 +\\
&\qquad 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3,\\
&g_1(\vec{x}) = -x_1 + 0.0193 x_3 \le 0,\\
&g_2(\vec{x}) = -x_3 + 0.00954 x_3 \le 0,\\
&g_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0,\\
&g_4(\vec{x}) = x_4 - 240 \le 0,\\
&0 \le x_1 \le 99,\\
&0 \le x_2 \le 99,\\
&10 \le x_3 \le 200,\\
&10 \le x_4 \le 200
\end{aligned}
\tag{7}
$$

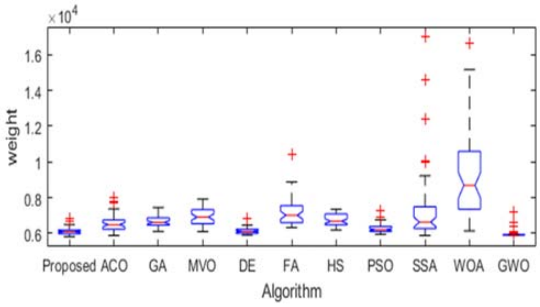

Fig. 9. Boxplot of optimization algorithms for pressure vessel.

Table IX presents the results of the comparison of the proposed algorithm with other algorithms for this problem. The results of Table IX show that the proposed algorithm has found the best values for the parameters of this problem with the lowest cost. The proposed algorithm has provided much better results than the basic DE, which indicates a good improvement of the DE algorithm. Fig.9 shows the boxplot of these results. It is also evident in this figure that the proposed algorithm has little variation domain, and the results are reliable.

TABLE IX RESULTS FOR PRESSURE VESSEL

Algorithms	Ts	Th	R	L	total cost
Proposed	**0.741008508178370**	**0.366280889951350**	**38.394223223450190**	**2.286564435936813e+02**	**5.824798828628441e+03**
ACO	0.777434038679288	0.384286048135101	40.281556408441084	2.005305344500567e+02	5.884078280730308e+03
SSA	0.837911788133658	0.414181115600703	43.415114710461370	1.610681292744691e+02	5.888208003957637e+03
GWO	0.779060201611711	0.385605947203811	40.356022228659130	1.994939256120291e+02	5.889669867585164e+03
DE	0.780528575775328	0.387266342572055	40.395530711566050	1.997447060478521e+02	5.917024284041992e+03
PSO	0.817546909041059	0.404113860738411	42.359943473795170	1.734229412412289e+02	5.956106814947694e+03
MVO	0.841994147840166	0.420230574312327	43.383507216212480	1.636621013054519e+02	6.104855009443729e+03
GA	0.885619998632570	0.437019418452654	45.651570746898430	1.370774062346848e+02	6.119592843410516e+03
WOA	0.814154631424362	0.430779780542335	41.336362536380830	1.863145086502679e+02	6.146038417556294e+03
HS	0.926043454960691	0.457433788247439	47.912021359273844	1.158333410458892e+02	6.195537242050121e+03
FA	0.921805255098330	0.455910591579237	47.761450214113170	1.222955539863427e+02	6.334605342004511e+03
BA	1.251347520363656	0.618558861660430	64.835889177315690	48.448017378547725	9.324375396912084e+03
ALO	2.744418974792293	0.815468737995623	55.139306233177734	67.086738176662070	2.056635223733777e+04
GSA	4.765878753715501	0.828904631477576	43.149588771131235	1.689427207932072e+02	5.596192960162536e+04

3) A three-bar truss design problem

In general, the problem of truss design is prevalent in the field of civil engineering. The purpose is to design a low-weight truss that does not violate the constraint. The most important issue in designing a truss is constraints that include stress, deflection, and buckling constraints. Figure 10 shows the structural parameters of this problem.

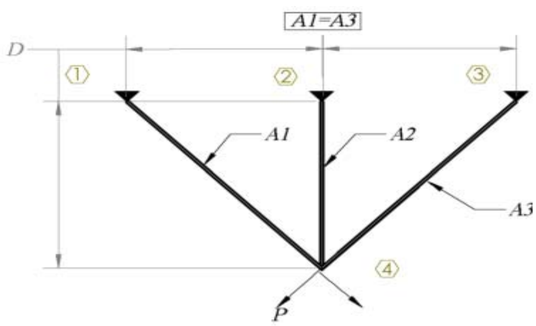

Fig. 10 Three bar-truss design problem [18].

The formula for this problem and its constraints are in the form of Equ. (8).

$$
\begin{aligned}
&Minimize: \quad f(A_1, A_2) = (2\sqrt{2A_1} + A_2) \times l \\
&Subject\,to: \\
&g_1 = \frac{\sqrt{2A_1} + A_2}{\sqrt{2}A_1^2 + 2A_1A_2} P - \sigma \le 0 \\
&g_2 = \frac{A_2}{\sqrt{2}A_1^2 + 2A_1A_2} P - \sigma \le 0 \\
&g_3 = \frac{1}{A_1 + \sqrt{2}A_2} P - \sigma \le 0 \\
&where \\
&0 \le A_1 \le 1 \quad and \quad 0 \le A_2 \le 1; \quad l = 100\,cm, \\
&P = 2KN/cm^2, \quad \sigma = 2KN/cm^2.
\end{aligned}
\tag{8}
$$

The results of the comparison of the performance of the proposed algorithm with other optimization algorithms are presented in Table X. In this problem, the proposed algorithm performs better than the other algorithms mentioned, and the proposed algorithm was able to provide the best values of the parameters with the least weight. As can be seen from the results in Table X, the results are very close together, and the proposed algorithm has been able to give better results.

TABLE X RESULTS FOR THREE-BAR TRUSS

Algorithm	d	D	weight
Proposed	**0.78867781353**	**0.40824071332**	**2.638958433817377e+02**
PSO	0.78867872161	0.40823814491	2.638958433859190e+02
FA	0.78876577397	0.40799208293	2.638958593120682e+02
GA	0.78863028258	0.40837516590	2.638958448549885e+02
ACO	0.78857445715	0.40853322470	2.638958609215823e+02
BA	0.78841089116	0.40899671736	2.638959467368605e+02
GSA	0.79249252715	0.39758796196	2.639095321943150e+02
DE	0.78868817252	0.40821145928	2.638958479415125e+02
HS	0.78705362428	0.41286129495	2.638985114510033e+02
MVO	0.78885344194	0.40774451450	2.638958987150073e+02
ALO	0.81245940423	0.34983962967	2.647821846365519e+02
WOA	0.78867256011	0.40825566377	2.638958525355147e+02
GWO	0.78865856636	0.40829801440	2.638961295684239e+02
SSA	0.78885469521	0.40774065336	2.638958471478859e+02

The boxplot of these results is shown in Fig.11. The domain of changes in most algorithms is low because of the low number of parameters, but the accuracy of the proposed algorithm is higher than the other algorithms and the Basic DE.

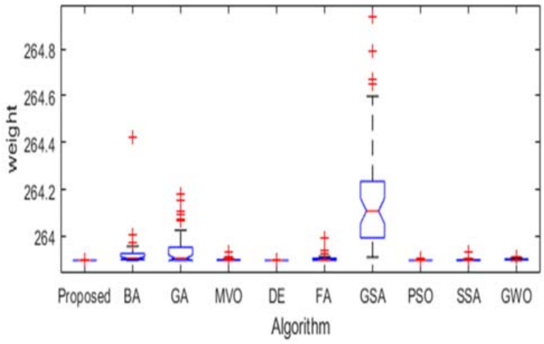

Fig. 11. Boxplot of optimization algorithms for three-bar truss

4) Economic load dispatch problem

The economic load dispatch problem is defined as minimizing the total operating cost of a power system while meeting the whole load plus transmission losses within the generator limits. Mathematically, the problem is outlined to minimize equation (9) subjected to the energy balance equation given by (10) and the inequality constraints given by Equ. (11).

$$F_i(P_i) = \sum_{i=1}^{NG} (a_i P_i^2 + b_i P_i + c_i) \tag{9}$$

$$\sum_{i=1}^{NG} P_i = P_D + P_L \tag{10}$$

$$P_{i\min} \le P_i \le P_{i\max} \, (i = 1, 2 \ldots NG) \tag{11}$$

Where
a_i, b_i and c_i are the cost coefficients
P_D is the load demand
P_i is the real power generation
P_L is the transmission power loss
NG is the number of generation buses.

One of the important, simple, but approximate methods of expressing transmission loss as a function of generator powers is through B-coefficients. The general form of the loss formula using B-coefficients is

$$P_i = \sum_{i=1}^{NG} \sum_{j=1}^{NG} P_i B_{ij} P_j \quad MW \tag{12}$$

Where

Pi and Pj are the real power generations at the i^{th} j^{th} buses respectively

B_{ij} are loss coefficients.

In a standard economic load dispatch problem, the input-output characteristics of a generator are approximated using quadratic functions, underneath the idea that the progressive cost curves of the units are monotonically increasing piecewise-linear functions. However, real input-output characteristics display higher-order nonlinearities and discontinuities due to valve–point loading in fossil fuel burning plants.

The generating units with multi-valve steam turbines exhibit a more significant variation in the fuel cost functions. The valve–point effects introduce ripples in the heat–rate curves. Mathematically operating cost is defined as:

$$F_i(P_i) = \sum_{i=1}^{NG} a_i P_i^2 + b_i P_i + c_i + |\, d_i \times \sin\{e_i \times (P_i^{\min} - P_i)\}\,|\,) \quad (13)$$

Where a_i, b_i, c_i, d_i and i are the cost coefficients of the ith unit.

In order to show the effectiveness of the proposed algorithm for the economic load dispatch problem, two power benchmark tests having standard IEEE bus systems have been taken into consideration. The proposed algorithm was performed 30 times with an initial population of 50 and 250 iterations on the economic load dispatch problem.

a. Test system I: 13-generating unit system without valve-point effect

The first test case consists of a 13-generating unit system without valve-point loading. The results of 13-generating unit systems are tested for load demand of 1800 MW and are shown in Table XI, and the effectiveness of the proposed algorithm for a 13-generating unit system is compared with Famous and new algorithms.

Corresponding analysis of results (Table XI) shows that the proposed algorithm has the lowest cost in terms of statistical average relative to other algorithms. To confirm the results, the boxplot of these 30 runs is shown in Fig. 12.

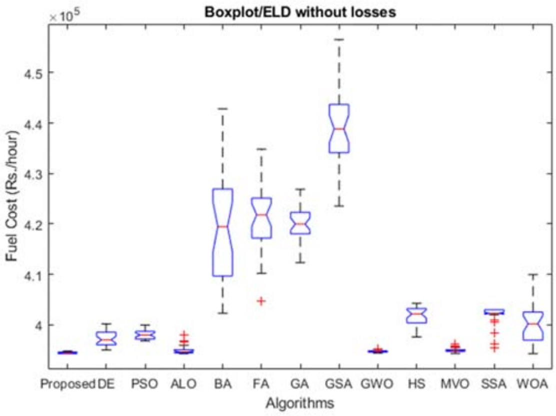

Fig. 12. Boxplot of the 13-generating unit system

TABLE XI RESULTS FOR ECONOMIC LOAD DISPATCH FOR A 13-GENERATING UNIT SYSTEM (LOAD DEMAND = 1800 MW)

algorithm	min	max	std	avg
Proposed	**3.942908360000000e+05**	**3.947708347473913e+05**	**1.702916262348927e+02**	**3.944528998165501e+05**
GWO	3.943755947652264e+05	3.953150723006313e+05	2.277009190496261e+02	3.947495936224741e+05
MVO	3.943079570069738e+05	3.962459245326403e+05	4.287871883177548e+02	3.949320390303100e+05
ALO	3.942945460790953e+05	3.979388786711827e+05	1.009983054859659e+03	3.949973550181657e+05
DE	3.950134788856796e+05	4.001816342752689e+05	1.472808687795597e+03	3.972839338063846e+05
PSO	3.968290219406239e+05	3.999403048419059e+05	9.220882528019836e+02	3.980576075108528e+05
WOA	3.942908362524503e+05	4.099378526143908e+05	4.205763758262005e+03	4.000378578427732e+05
HS	3.975843848286539e+05	4.042595044850244e+05	1.712676986482858e+03	4.017708384618684e+05
SSA	3.953518834700992e+05	4.029978440000000e+05	1.916308443471383e+03	4.019297078929522e+05
BA	4.022771094724992e+05	4.428226863427925e+05	1.260470203951983e+04	4.188199079380141e+05
GA	4.123058029373971e+05	4.268616332165159e+05	3.085039990981851e+03	4.202360516049843e+05
FA	4.047544719345829e+05	4.348094421613113e+05	6.399301093928142e+03	4.208275325399591e+05
GSA	4.235363270478695e+05	4.565316587297490e+05	7.099053517237799e+03	4.390921865866760e+05

As shown in Fig.12, the proposed method has the least variance, which indicates the stability of the proposed algorithm. Fig. 13 shows the convergence curve of the optimization algorithms for this system.

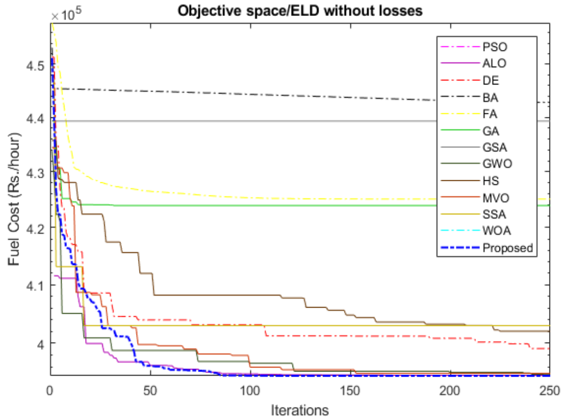

Fig. 13. Convergence curve of the 13-generating unit system

b. Test system II: 40-generating unit system considering the valve-point effect

The second test system, which consists of a 40-generating unit system, is tested for load demand of 10500 MW. The Valve-point effect is taken into consideration, but transmission losses are neglected while calculating the optimal fuel cost. The results of 40-generating unit systems are shown in Table XII. The results of Table XII show that the proposed algorithm performs better in terms of statistical (minimum, maximum, standard deviation, and average) relative to other algorithms. This indicates that the proposed algorithm still performs better than other compared algorithms by applying more constraints such as valve-point and increasing the problem dimension. The base DE algorithm ranks second, but the average value of the proposed algorithm is better than the minimum value of the base DE algorithm. To confirm the results, the boxplot of these 30 runs is shown in Fig. 14.

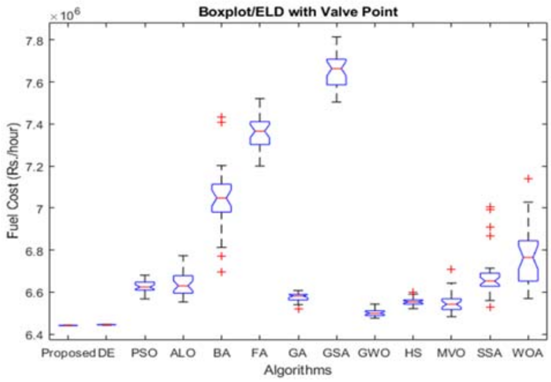

Fig. 14. Boxplot of the 40-generating unit system

As shown in Fig.14, the proposed method has the least variance.

Fig. 15 shows the convergence curve of the optimization algorithms for this system. Fig. 15 shows that the proposed algorithm, despite considering the valve point, still achieves the optimal solution in a fewer number of iterations.

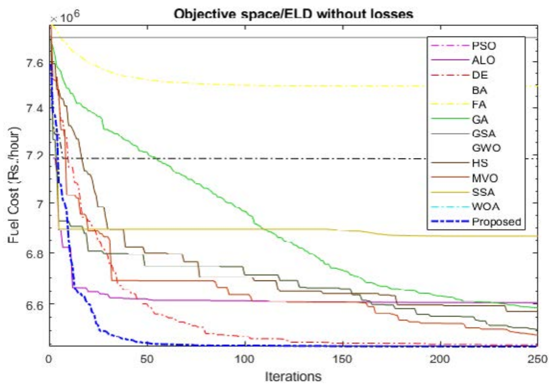

Fig. 15. Convergence curve of the 40-generating unit system

TABLE XII RESULTS FOR 40-GENERATING UNIT SYSTEM CONSIDERING VALVE-POINT EFFECT (LOAD DEMAND = 10500 MW)

algorithm	min	max	std	avg
Proposed	6.440254870576086e+06	6.443429084201531e+06	9.267728034246223e+02	6.441317109662098e+06
DE	6.442694793383291e+06	6.447102496675514e+06	1.065898382895214e+03	6.445087632008978e+06
GWO	6.476249343497660e+06	6.543482225889794e+06	1.673905935216343e+04	6.502739076335037e+06
MVO	6.483471202451115e+06	6.708437170758968e+06	5.021813748181564e+04	6.551313544530138e+06
HS	6.521737675525253e+06	6.598121250377092e+06	1.741968228283060e+04	6.553795292600089e+06
GA	6.520351087295961e+06	6.607557728287661e+06	2.079250768996690e+04	6.575688950935189e+06
PSO	6.568017575171378e+06	6.681006511252836e+06	2.706445836273727e+04	6.629674343475549e+06
ALO	6.553449946566720e+06	6.773381467996010e+06	5.714175478709766e+04	6.638563252619594e+06
SSA	6.528102734684937e+06	7.006878873392217e+06	1.282970249264827e+05	6.692945654985365e+06
WOA	6.569189111112373e+06	7.139375955368181e+06	1.502834340571168e+05	6.764319869304523e+06
BA	6.695914132028671e+06	7.433198314429556e+06	1.592634081638326e+05	7.045047474057742e+06
FA	7.200410107332142e+06	7.521353119082810e+06	8.116620401192234e+04	7.364066493960878e+06
GSA	7.504839165329872e+06	7.814783285308058e+06	7.927095429937584e+04	7.653910624228362e+06

V. CONCLUSIONS

In this paper, we propose a new hybrid algorithm that combines the differential evolution algorithm and our proposed formula. Optimization algorithms usually include two phases of exploration and exploitation. The differential evolution algorithm performs well in the exploration phase, but the exploitation phase is weak. In fact, for the greater effectiveness of the exploitation phase, we added a formula to the differential evolution algorithm. To evaluate the proposed algorithm in terms of exploration and exploitation, 19 test functions including seven unimodal test functions to evaluate algorithm exploitation, six multimodal test functions to evaluate algorithm exploration, and six composite test functions to evaluate the escape from local optimal of the algorithm, were used. The results showed that our proposed algorithm has good performance and competitive performance compared to other famous and novel optimization algorithms. Also, to evaluate the algorithm in unknown search spaces, the proposed algorithm was applied to several well-known engineering design problems which result show the high performance of the proposed algorithm in solving problems with unknown searching spaces.

REFERENCES

1. A. R. Simpson, G. C. Dandy, and L. J. Murphy, "Genetic Algorithms Compared to Other Techniques for Pipe Optimization," J. Water Resour. Plan. Manag., vol. 120, no. 4, pp. 423–443, Jul. 1994, DOI: 10.1061/(ASCE)0733-9496(1994)120:4(423).

2. J. C. Spall, Introduction to Stochastic Search and Optimization. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2003.

3. I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," Inf. Sci. (NY)., vol. 237, pp. 82–117, Jul. 2013, DOI: 10.1016/j.ins.2013.02.041.

4. J. A. Parejo, A. Ruiz-Cortés, S. Lozano, and P. Fernandez, "Metaheuristic optimization frameworks: a survey and benchmarking," Soft Comput., vol. 16, no. 3, pp. 527–561, Mar. 2012, DOI: 10.1007/s00500-011-0754-8.

5. M. Dorigo and T. Stützle, Ant Colony Optimization. Scituate, MA, USA: Bradford Company, 2004.

6. E.-G. Talbi, Metaheuristics: From Design to Implementation. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2009.

7. M. Mafarja et al., "Evolutionary Population Dynamics and Grasshopper Optimization approaches for feature selection problems," Knowledge-Based Syst., vol. 145, pp. 25–45, Apr. 2018, DOI: 10.1016/j.knosys.2017.12.037.

8. A. A. Heidari, R. Ali Abbaspour, and A. Rezaee Jordehi, "An efficient chaotic water cycle algorithm for optimization tasks," Neural Comput. Appl., vol. 28, no. 1, pp. 57–85, Jan. 2017, DOI: 10.1007/s00521-015-2037-2.

9. I. Aljarah, M. Mafarja, A. A. Heidari, H. Faris, Y. Zhang, and S. Mirjalili, "Asynchronous accelerating multi-leader salp chains for feature selection," Appl. Soft Comput., vol. 71, pp. 964–979, Oct. 2018, DOI: 10.1016/j.asoc.2018.07.040.

10. M. Mafarja et al., "Binary dragonfly optimization for feature selection using time-varying transfer functions," Knowledge-Based Syst., vol. 161, pp. 185–204, Dec. 2018, DOI: 10.1016/j.knosys.2018.08.003.

11. J. H. . Holland, "Genetic Algorithms understand Genetic Algorithms," Surprise 96, vol. 1, no. 1, pp. 12–15, 1967, DOI: 10.2307/24939139.

12. R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 2002, pp. 39–43, DOI: 10.1109/MHS.1995.494215.

13. A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed Optimization by Ant Colonies," in European Conference on artificial life, 1991, vol. 142, pp. 134–142.

14. R. Storn and K. Price, "Differential Evolution- A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces," Tech. Rep. TR-95-012, vol. 11, no. 4, pp. 1–12, 1995, DOI: https://doi.org/10.1023/A:1008202821328.

15. D. Manjarres et al., "A survey on applications of the harmony search algorithm," Eng. Appl. Artif. Intell., vol. 26, no. 8, pp. 1818–1831, Sep. 2013, DOI: 10.1016/j.engappai.2013.05.008.

16. X.-S. Yang, "Firefly Algorithm, Lévy Flights, and Global Optimization," in Research and Development in Intelligent Systems XXVI, London: Springer London, 2010, pp. 209–218.

17. X. Yang and A. Hossein Gandomi, "Bat algorithm: a novel approach for global engineering optimization," Eng. Comput., vol. 29, no. 5, pp. 464–483, Jul. 2012, DOI: 10.1108/02644401211235834.

18. S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems," Adv. Eng. Softw., vol. 114, pp. 163–191, Dec. 2017, DOI: 10.1016/j.advengsoft.2017.07.002.

19. S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," Adv. Eng. Softw., vol. 69, pp. 46–61, Mar. 2014, DOI: 10.1016/j.advengsoft.2013.12.007.

20. S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," Adv. Eng. Softw., vol. 95, pp. 51–67, May 2016, DOI: 10.1016/j.advengsoft.2016.01.008.

21. E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A Gravitational Search Algorithm," Inf. Sci. (NY)., vol. 179, no. 13, pp. 2232–2248, Jun. 2009, DOI: 10.1016/j.ins.2009.03.004.

22. S. Mirjalili, S. M. Mirjalili, and A. Hatamlou, "Multi-Verse Optimizer: a nature-inspired algorithm for global optimization," Neural Comput. Appl., vol. 27, no. 2, pp. 495–513, Feb. 2016, DOI: 10.1007/s00521-015-1870-7.

23. S. Mirjalili, "The Ant Lion Optimizer," Adv. Eng. Softw., vol. 83, pp. 80–98, 2015, DOI: https://doi.org/10.1016/j.advengsoft.2015.01.010.

24. Anita, A. Yadav, and N. Kumar, "Artificial electric field algorithm for engineering optimization problems," Expert Syst. Appl., vol. 149, p. 113308, Jul. 2020, DOI: 10.1016/j.eswa.2020.113308.

25. E. H. Houssein, M. R. Saad, F. A. Hashim, H. Shaban, and M. Hassaballah, "Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems," Eng. Appl. Artif. Intell., vol. 94, p. 103731, Sep. 2020, DOI: 10.1016/j.engappai.2020.103731.

26. S. H. Samareh Moosavi and V. K. Bardsiri, "Poor and rich optimization algorithm: A new human-based and multi populations algorithm," Eng. Appl. Artif. Intell., vol. 86, pp. 165–181, Nov. 2019, DOI: 10.1016/j.engappai.2019.08.025.

27. S. Kaur, L. K. Awasthi, A. L. Sangal, and G. Dhiman, "Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization," Eng. Appl. Artif. Intell., vol. 90, p. 103541, Apr. 2020, DOI: 10.1016/j.engappai.2020.103541.

28. D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 67–82, Apr. 1997, DOI: 10.1109/4235.585893.

29. M. Dehghani, Z. Montazeri, O. P. Malik, A. Ehsanifar, and A. Dehghani, "OSA: Orientation Search Algorithm," Int. J. Ind. Electron. Control Optim., vol. 2, no. 2, pp. 99–112,

2019, DOI: 10.22111/ieco.2018.26308.1072.

30. A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine Predators Algorithm: A nature-inspired metaheuristic," Expert Syst. Appl., vol. 152, p. 113377, Aug. 2020, DOI: 10.1016/j.eswa.2020.113377.

31. P. Suganthan et al., "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Nat. Comput., vol. 341–357, Jan. 2005.

32. P. Civicioglu, E. Besdok, M. A. Gunen, and U. H. Atasever, "Weighted differential evolution algorithm for numerical function optimization: a comparative study with cuckoo search, artificial bee colony, adaptive differential evolution, and backtracking search optimization algorithms," Neural Comput. Appl., 2018, DOI: 10.1007/s00521-018-3822-5.

33. P. Civicioglu and E. Besdok, "Bernstein-search differential evolution algorithm for numerical function optimization," Expert Syst. Appl., vol. 138, p. 112831, Dec. 2019, DOI: 10.1016/j.eswa.2019.112831.

34. J. E. V. Ferreira, M. T. S. Pinheiro, W. R. S. dos Santos, and R. da S. Maia, "Graphical representation of chemical periodicity of main elements through boxplot," Educ. Química, vol. 27, no. 3, pp. 209–216, Jul. 2016, DOI: 10.1016/j.eq.2016.04.007.

35. F. van den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," Inf. Sci. (NY)., vol. 176, no. 8, pp. 937–971, 2006, DOI: https://doi.org/10.1016/j.ins.2005.02.003.

36. J. S. Arora, Introduction to Optimum Design. Elsevier, 2017.

37. S. Khalilpourazari and S. Khalilpourazary, "An efficient hybrid algorithm based on Water Cycle and Moth-Flame Optimization algorithms for solving numerical and constrained engineering optimization problems," Soft Comput., vol. 23, no. 5, pp. 1699–1722, Mar. 2019, DOI: 10.1007/s00500-017-2894-y.

38. X. Han, Q. Liu, H. Wang, and L. Wang, "Novel fruit fly optimization algorithm with trend search and co-evolution," Knowledge-Based Syst., vol. 141, pp. 1–17, Feb. 2018, DOI: 10.1016/j.knosys.2017.11.001.